Títol: Serveis de base de dades de gestió docent

Volum: 1/1

Alumne: Fernando Peña Ardanuy

Director/Ponent: Enric Xavier Martin i Rull

Departament: ESAII

Data: 01-06-2009

DADES DEL PROJECTE

Títol del Projecte: Serveis de base de dades de gestió docent

Nom de l'estudiant: Fernando Peña Ardanuy
Titulació: Enginyeria informàtica
Crèdits: 37.5
Director/Ponent: Enric Xavier Martin Rull
Departament: ESAII

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Antonio Benito Martínez Velasco

Vocal: Guillermo González Casado

Secretari: Enric Xavier Martin i Rull

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

1.- Introducción

1.1.- Contexto del proyecto

El proyecto "Serveis de base de dades de gestió docent" se desarrolla para el departamento de Enginyeria de Sistemas, Automàtica i Informàtica Industrial, ESAII de la Universitat Politècnica de Catalunya, UPC. Este proyecto surge de la necesidad de actualización del sistema de gestión de la docencia del departamento de ESAII. Se busca que el nuevo sistema sea accesible, usable, fiable y seguro.

La principal tarea del nuevo sistema de gestión consiste en poder realizar encargos de asignaturas a un departamento y asignar horas lectivas a profesores, de los grupos de las asignaturas encargadas al departamento.

También se debe mostrar y permitir la modificación de toda la información sobre profesores, encargos, asignaciones, departamentos, centros, secciones, titulaciones y horarios de los profesores, así como la creación y consulta de balances por secciones, profesores y departamentos.

1.1.1.- Problema

A la hora de plantear la situación del nuevo sistema de gestión se marcaron unos requisitos mínimos que este debía cumplir:

- Completa funcionalidad.
- Fácil acceso.
- Fiabilidad.
- Usabilidad.
- Minimización del impacto en los usuarios.
- Seguridad en los accesos y el contenido.

Estos requisitos marcaron las principales características del proyecto. A continuación se explica como cada uno de estos requisitos ha afectado a la solución final y que se ha hecho para cumplirlos.

El requisito de completa funcionalidad implica que todo lo que se hiciera con el anterior método de gestión, también se debería poder hacer con el nuevo. Es decir, se debe permitir y controlar la alta, baja y modificación de todos y cada uno de los elementos que intervienen en el proceso de gestión de la docencia del departamento: secciones, centros, profesores, tipos de contrato, cargos de los profesores y asignaturas. Además de permitir realizar el punto principal de la gestión, que es la asignación de horas de docencia a los profesores y la definición de los horarios de cada grupo.

La facilidad de acceso del sistema debe permitir que, cada usuario, en este caso cada profesor, que deseara acceder al sistema, debería poder hacerlo sin mayores complicaciones y, a ser posible, sin tener que instalar nada, para facilitar el acceso desde cualquier punto. Por lo que se decidió que el método de acceso sería a través de un portal Web , que permitiría el acceso al sistema desde cualquier punto con conexión a Internet.

El sistema debe ser fiable, lo que implica que los datos almacenados deben ser accesibles y lo más estables posibles, por lo que se decidió que se almacenarían en una base de datos, lo que permite centralizar los datos y crear copias de seguridad para poder reponerlos en caso de emergencia.

La usabilidad del sistema se debe garantizar, ya que un sistema complejo de usar es un sistema no usado y uno de los objetivos de este sistema es reemplazar al anterior, por lo que se ha de garantizar la usabilidad con tal de convencer al usuario de que este sistema es mejor.

Por último se debe minimizar el impacto en los usuarios. Es decir, se debe permitir que los usuarios puedan seguir ejecutando los procesos de gestión tal y como los hacían antes o, de una manera similar. En este aspecto se tienen en cuenta factores como el uso de herramientas que los usuarios ya usaran con anterioridad, en este caso la hoja de cálculo Excel, ya que es lo que se utilizaba para generar las asignaciones y permitiría conservar la forma de trabajar de los usuarios. Se ha de tener en cuenta que el uso del formato xls, propio de los archivos de Excel, no obliga a usar Microsoft Excel, ya que xls es un formato libre y existe soluciones libres, como Open Office, que son capaces de manejarlo.

Con tal de poder asumir estos requisitos, el sistema de gestión se ha implementado utilizando la interfaz de Java Servlets, que permite la creación y mantenimiento de una Web, residente en un servidor Apache Tomcat, el sistema gestor de bases de datos MySQL, que permitirá implementar la persistencia de datos y las librerías Jakarta POI de la fundación Apache, que permiten la lectura y edición de archivos de formato xls a través de Java.

1.1.2.- Objetivos

Como se ha comentado en la introducción, el objetivo principal es la creación de un sistema de gestión de la docencia que permita sustituir al usado actualmente.

A continuación se especifican tanto los requisitos funcionales como los no funcionales que se establecieron durante la fase de toma de requisitos del proyecto, de acuerdo a las necesidades de los usuarios del sistema.

Requisitos funcionales

1. Definición y modificación de departamentos, centros, secciones, tipos de contrato y titulaciones.
   Un departamento está en constante evolución, al igual que una universidad, es por ello que el sistema debe permitir que se añadan nuevos elementos que permitan que el sistema se mantenga al día. Además también cabe la posibilidad de que el sistema se extienda a otros departamentos diferentes al departamento para el que se diseñó, por lo que el sistema se ha de diseñar pensando en esta posibilidad.

2. Alta, baja y modificación de profesores y sus datos personales, así como sus asignaciones a departamentos y centros.
   El personal de un departamento no es una cosa fija, con el tiempo hay altas y bajas de nuevo personal, o cambios de sección o de departamento, por lo que el sistema debe contemplar esta situación e implementarla.

3. Incorporación de encargos de asignaturas y grupos al sistema
   La universidad realiza encargos de asignaturas a impartir y grupos a crear a los departamentos. El sistema debe gestionar estos encargos e incorporarlos a sus base de datos para tenerlos en cuenta a la hora de adjudicar horas lectivas a profesores.

4. Asignación de horas lectivas a profesores.
   El sistema debe implementar un método para realizar y consultar la asignación de horas lectivas a profesores, así como mantener su persistencia.

5. Definición y consulta de horarios de profesores
   El sistema debe permitir que cada uno de los profesores sea capaz de, dados cada uno de los grupos en los que tiene que impartir clases, definir el horario de estos y consultarlo.

6. Generación de informes y balances departamentales
   El sistema debe ser capaz de crear informes departamentales que muestren el total de puntos que cada profesor acumula en cada una de las secciones del departamento. Además también se deben crear balances que muestren la capacidad de un departamento y su encargo, para así detectar posibles sobrecargas a tiempo.

Requisitos no funcionales

1. La construcción y el mantenimiento del sistema deben representar el mínimo coste posible y , a ser posible, el programa resultante debe ser de código abierto, lo que en una comunidad universitaria facilita la distribución y comunicación de resultados del mismo.
   Para cumplir con este requisito, se optó por usar una serie de tecnologías, de código abierto, indicadas a continuación, junto con las referencias a sus licencias.
   - **JRE 6** : Java Runtime Environment 6, cuya licencia, disponible en *http://java.com/es/download/license.jsp*, permite el uso y la distribución de código basado en la plataforma Java libremente.
   - **MySQL**: Sistema gestor de bases de datos de código abierto distribuido bajo la licencia GPL (General Public License), disponible en *www.fsf.org/licensing/licenses/gpl.html*, que permite el uso y la libre distribución del código.
   - **Servidor Apache Tomcat**: Gestor de contenidos, distribuido por la Apache Software Foundation, bajo la Apache License v2.0, disponible en *www.apache.org/licenses/LICENSE-2.0*, que permite el uso, la modificación y distribución del código.
   - **Proyecto Apache POI**: Librería que permite la comunicación con herramientas Microsoft Office, distibuida por la Apache Software Foundation, bajo la Apache License v2.0, disponible en *www.apache.org/licenses/LICENSE-2.0*, que permite el uso, la modificación y distribución del código.

2. El impacto, del uso del nuevo sistema de gestión, en los usuarios se debe minimizar.
   El personal del departamento estaba acostumbrado a realizar los procesos de gestión usando la herramienta Microsoft Excel, es por eso que se ha optado por utilizar las librerías del proyecto Apache POI, que permiten la comunicación entre archivos xls y Java, para que los usuarios puedan continuar trabajando como lo venían haciendo.

3. El sistema debe tener la mayor accesibilidad posible
   Considerando este requisito, se desechó la idea de crear una aplicación de gestión y se optó por la creación de una plataforma Web, ya que lo único necesario para acceder a esta es una conexión a Internet y un navegador compatible (Mozilla Firefox, Safari, Internet Explorer…).

1.2.- Propuesta de solución

Una vez establecidos y analizados los objetivos del proyecto, se buscó una solución de diseño que fuera capaz de cumplir todos los requisitos, ya fueran funcionales o no funcionales. Teniendo en cuenta que si a alguno de estos no se les podía dar soporte, siempre habría preferencia de los requisitos funcionales en detrimento de los no funcionales.

El diseño obtenido de este proceso de toma y análisis de requisitos consistió en la creación de una plataforma Web, basada en la interfaz Java servlet, junto con el apoyo de una base de datos implementada en MySQL, además del uso de unas librerías Apache POI, creadas por la fundación Apache, que permiten la lectura y modificación de archivos de Microsoft Office usando el lenguaje de programación Java.

Este diseño permite cumplir con todos los requisitos, ya que toda la funcionalidad requerida por el sistema se basa en un adecuado diseño de la base de datos y la capacidad de representar esta información a través de la plataforma Web. Además el uso de las librerías Apache POI, junto con la interfaz de Java Servlets permiten, que el usuario pueda descargarse archivos de hoja de cálculo xls, trabajar con ellos y después subirlos al servidor para llevar a cabo todos los cambios allí indicados.

Todos los programas de terceros sugeridos en el diseño están distribuidos bajo licencias de código abierto, ya sean licencias GPL o licencias Apache, lo que permite la utilización de los programas sin tener que pagar una licencia adicional de uso.

La minimización del impacto en los usuarios se consigue usando las librerías Apache POI, que permiten trabajar con archivos xls.

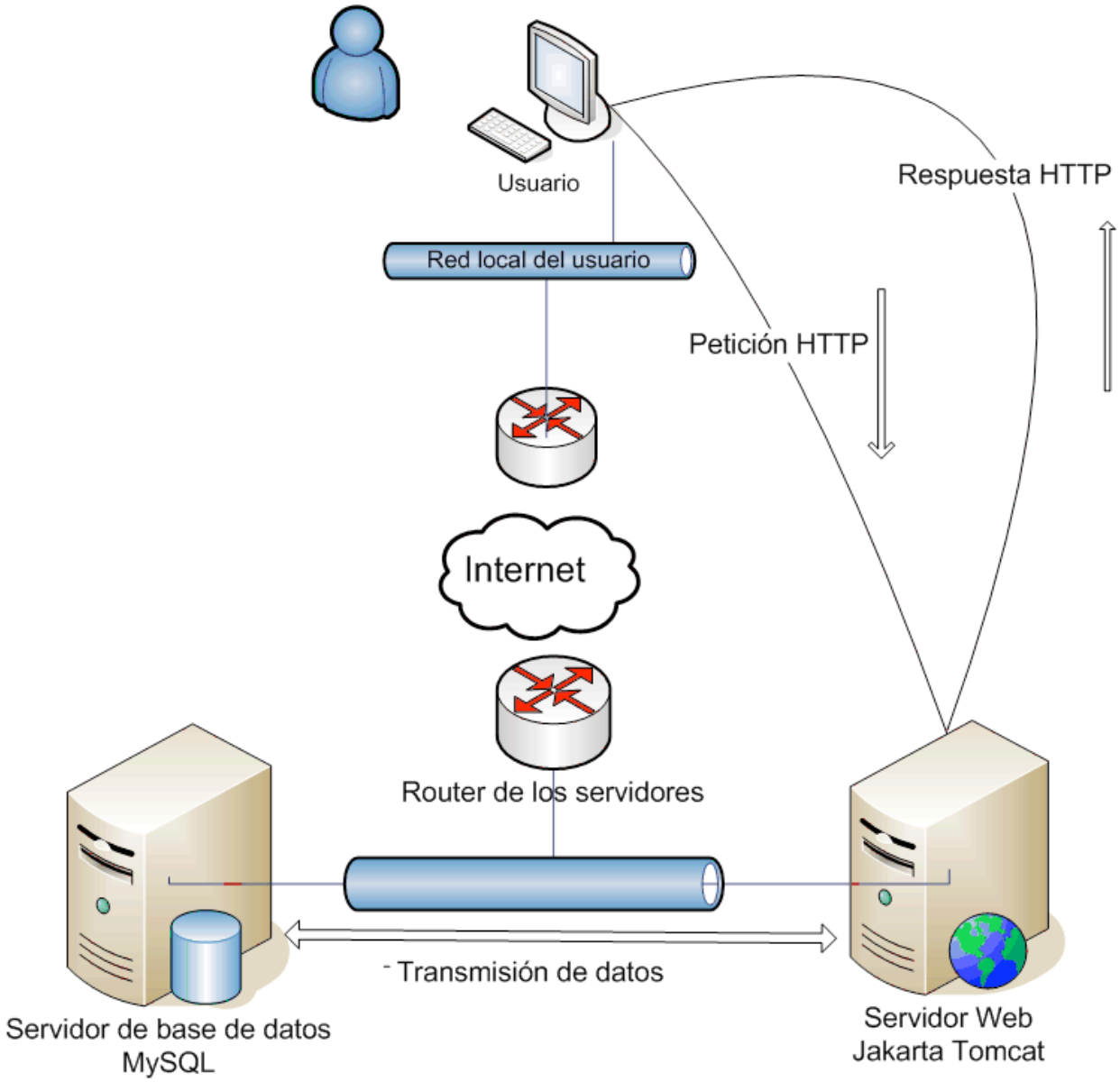
Por último, la creación de una plataforma Web, a través de un Servlet, permite que la consulta de información sea rápida y sencilla, además este tipo de plataformas también ofrecen una interfaz de descarga y subida de archivos, que en este caso servirá para descargar los datos en formato xls y poder actualizarlos mediante la subida de archivos de este tipo.

Este análisis previo del diseño demuestra que cumple con todos los requisitos tanto funcionales como no funcionales, por lo que resulta adecuado para la construcción del nuevo sistema de gestión de la docencia.

1.2.1.- Arquitectura de la solución

A continuación se muestra un diagrama de la arquitectura del diseño preliminar del sistema de gestión, donde se puede observar la dimensión de este de una manera global.

Lo representado es toda la infraestructura necesaria, en el entorno de producción, para que el sistema diseñado funcione. Además de esta infraestructura, el usuario necesitará disponer de una conexión a Internet, un navegador de Internet (Firefox, Safari, Explorer…) y de un gestor de hojas de cálculo (Microsoft Office, Open Office, …), para poder leer y escribir los archivos descargados del sistema.



Tal y como se puede observar en el diseño realizado, la comunicación básica con el sistema se realiza mediante mensajes HTTP.

La comunicación es iniciada por el usuario, cuando accede a la página principal de la Web, o cuando rellena un formulario de un apartado de la Web y envía la información que haya introducido, o los archivos que quiera subir al servidor. Esta información se envía al servidor en forma de petición http, con toda la información codificada en esta.

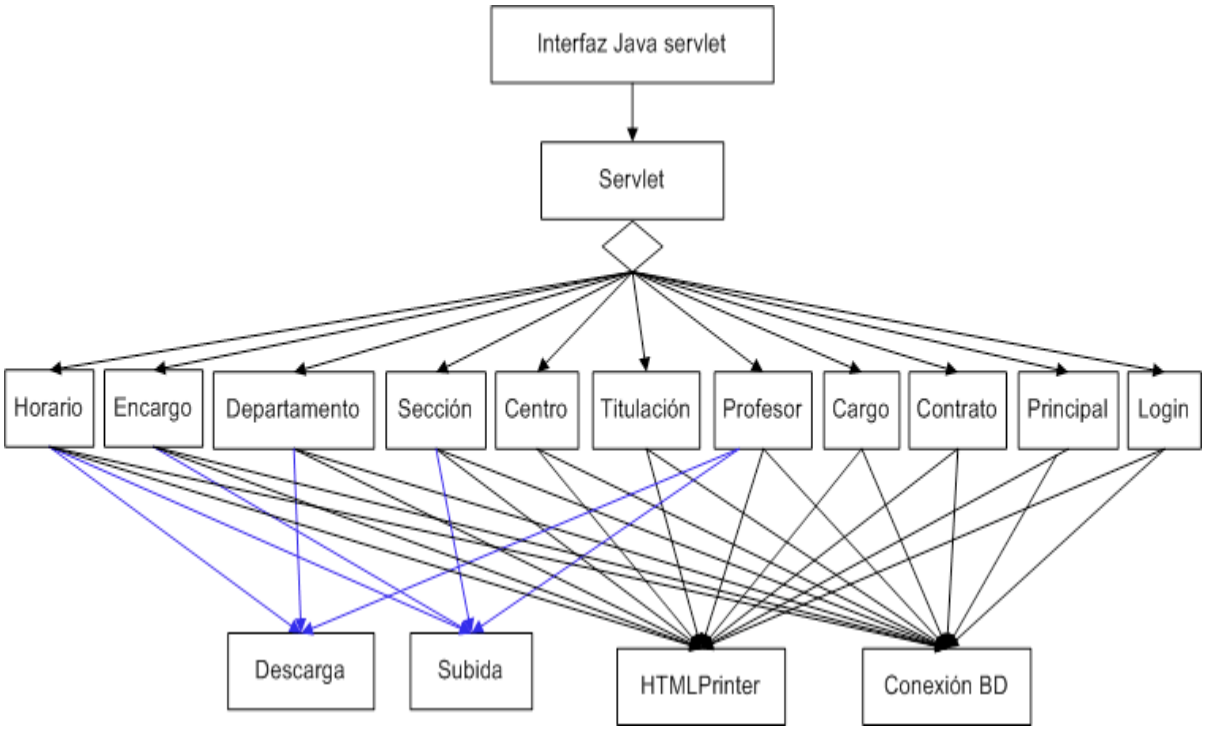
Tras recibir la petición HTTP, el servidor Web la procesa y se comunica con el servidor de bases de datos MySQL para realizar los intercambios de información necesarios.

La recepción de dicha petición, por parte del servidor, inicia el procesado de esta. El servidor le cede el control al Servlet y este, si es necesario, se comunica con la base de datos para intercambiar información.

Por último, una vez consultada la base de datos, el Servlet genera la respuesta adecuada y se la envía al usuario, ya sea en forma de página Web o de archivo a descargar.

1.2.2.- Estructura del servlet

A continuación se muestra un diagrama que representa la estructura de clases en las que se ha dividido el sistema construido, implementado sobre la interfaz de Java Servlet.



La parte principal de la implementación es el servlet, que hereda directamente las funciones y atributos de la interfaz servlet de java. Es este servlet el que procesa la petición HTTP y llama a la clase correspondiente (profesor, departamento, …), que será la que genere la respuesta conveniente.

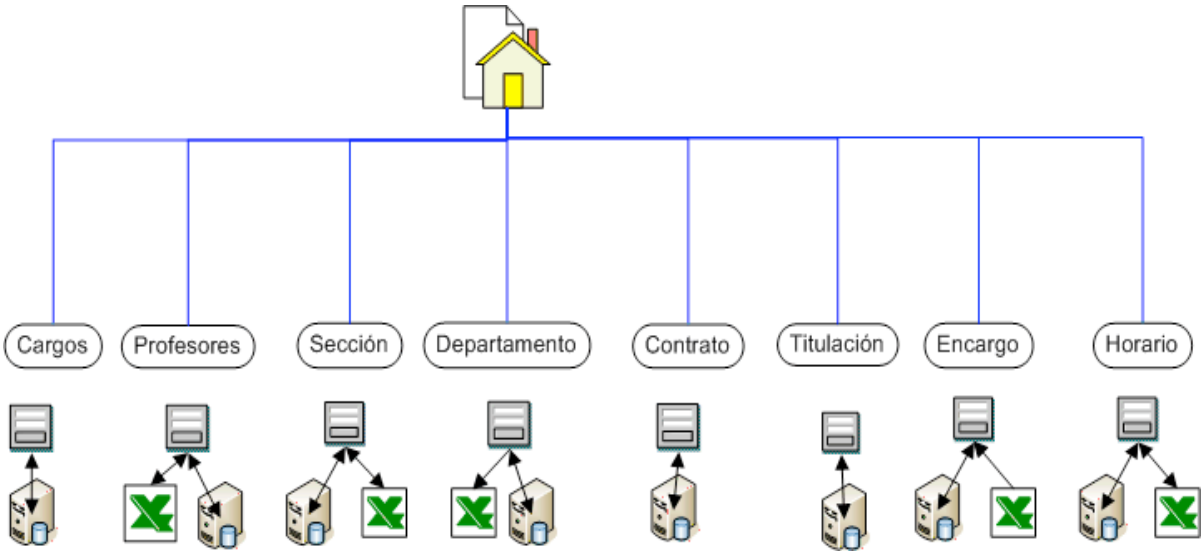
Estas clases secundarias, son las que se procesan el contenido de la petición HTTP y analizan la información introducida por el usuario. Prepara las consultas necesarias para la base de datos y mantiene la conexión con la misma. Así como se encarga de crear e interpretar los archivos xls que se usarán para la comunicación con el usuario.

La clase de conexión con la base de datos simplemente se encarga de gestionar la conexión y enviar las consultas a la base de datos.

Por último está la clase HTMLPrinter, que es la que genera el código HTML que representa la página Web que el usuario verá, codificando en esta la información recopilada de la base de datos.

1.2.3.- Mapa Web

A continuación se muestra un mapa navegacional de la plataforma Web que se ha desarrollado.



Este mapa Web permite observar las diferentes pantallas a las que se puede acceder desde la página de inicio, así como comprobar qué se puede hacer desde cada pantalla.

En este caso cada una de las pantallas es posible comunicarse con la base de datos a través de un formulario. Dicha comunicación es bidireccional, es decir, desde cada pantalla se puede tanto introducir información nueva como acceder a información ya existente en el sistema.

Por otra parte, en las pantallas de profesores, departamento, sección, encargo y horario existen las opciones de comunicación con archivos en formato xls, para consultar información del sistema, así como subirlos al servidor para cargar nuevas configuraciones en el sistema.

2.- Descripción de las tecnologías empleadas

En este apartado se describen las diferentes tecnologías empleadas para la realización del proyecto, concretamente se explica, brevemente, en qué consisten y se enuncian sus principales características.

2.1.- JAVA

El proyecto se ha implementado utilizando el lenguaje de programación Java. A continuación se ofrece una introducción a los principios del lenguaje, una breve historia del mismo, sus principales características, su uso y la interfaz de servlets, que es por lo que se ha elegido Java para crear el software.

2.1.1.- Introducción

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a mediados de los 90. Actualmente es uno de los lenguajes más utilizados, especialmente en entornos Web, debido a que es independiente de la plataforma sobre la que se ejecuta o compila.
Java se distribuye sobre una licencia que permite libertad de uso, creación y prueba de programas basados en la plataforma Java, siempre y cuando las partes centrales de esta plataforma no sean modificadas.

La sintaxis de Java está basada en la de C++, por lo que estos dos lenguajes comparten gran parte de su sintaxis. Esto se hizo porque en el momento en que se diseñó Java, el máximo exponente de la orientación a objetos era C++. Sin embargo el objetivo principal de Java, la independencia de la plataforma, que es lo que justifica su existencia, es diferente al de C++ y es lo que lo hace único.

2.1.2.- Historia

El proyecto Java comenzó a gestarse en 1991. En aquel momento el objetivo de Sun era crear una nueva tecnología que permitiera programar las nuevas generaciones de electrodomésticos inteligentes. Primero comenzaron trabajando con C++, pero los requisitos que este necesitaba para funcionar, superaban ampliamente las limitaciones técnicas de los sistemas que se pretendían controlar. Además también se encontraron con que el hecho de programar en C++, obligaba al programador a controlar manualmente el uso y liberación de memoria, lo que desembocaba en gran cantidad de errores. Otro problema con el que se encontraron era la falta de portabilidad y capacidad de paralelismo que el C++ ofrecía.

Una vez detectados todos estos problemas, el equipo de desarrollo se decantó por la creación de una nueva plataforma que los programas realizados sobre dicha plataforma pudieran ser portados de un dispositivo a otro.

Una vez definidos estos objetivos hubo dos opciones, la de James Gosling, que quería implementar la nueva plataforma como una reforma de C++, llamada C++ ++ o la opción de Bill Joy, que consistía en tomar como base la sintaxis de Mesa y C y sobre ello crear

un lenguaje de programación completamente nuevo que siguiera el paradigma de la orientación a objetos. Al producto de este desarrollo se le llamo Oak, en honor a un roble que James Gosling tenía en su jardín.

En sus primeras versiones la plataforma Oak generó grandes expectativas, e incluso, en 1992, se creó una nueva empresa, subsidiaria de Sun Microsystems, que se dedicaba exclusivamente a desarrollar Oak, también conocido por aquel entonces como proyecto Green.

Una de las primeras aplicaciones que se desarrollaron fue una interfaz altamente interactiva para una empresa de televisión por cable. Sin embargo esta empresa decidió que el uso de la nueva interfaz daba demasiada libertad al usuario y decidió cancelar el proyecto. Decisión que estancó el desarrollo de la plataforma y obligó a disolver la empresa creada y a que Sun volviera a asumir el control del proyecto.

Tras un periodo de letargo, se retomó el proyecto Green, pero esta vez orientándolo a la recién nacida World Wide Web, que ofrecía al usuario una libertad similar a la de la interfaz de televisión por cable creada en 1991.

Fue en 1994, cuando al intentar registrar la marca Oak, descubrieron que ya estaba registrada y le dieron el nombre de Java, aunque no se sabe a ciencia cierta de donde viene el nombre de Java, una corriente de opinión dice que es por el café que servían en la cafetería en la que se reunían los programadores, y que por eso el símbolo es una taza de café, y otra que es una distribución aleatoria de las iniciales de sus apellidos.

No obstante no fue hasta el 23 de mayo de 1995, cuando la primera versión de Java salió a la luz, junto con la noticia de que el primer navegador Netscape incluiría soporte a Java, lo que aumentó la popularidad de Java y contribuyó a la creación del grupo de desarrollo JavaSoft dentro de Sun en enero de 1996.

El primer kit de desarrollo de Java, JDK 1.0, salió al mercado el 23 de enero de 1996, junto con la librería estándar de Java y numerosos paquetes adicionales.

Desde ese momento Java se ha ido desarrollando y cada vez han ido incorporando más interfaces de programación, tales como Swing para la creación de interfaces gráficas, Servlet para la programación de plataformas web y JDBC para la conexión con bases de datos.

En 1998 salió a la luz la versión Java 2 y en 1999 se liberó su código. Desde entonces, prácticamente cada 2 años, se ha creado una nueva versión de la plataforma J2SE (Java 2 Standard Edition).

En noviembre de 2006 Sun publicó gran parte del código de Java como código abierto bajo una licencia GPL (General Public License). Hasta que en mayo de 2007 se acabó de liberar todo el código del núcleo de Java, excepto una pequeña porción que no está bajo el copyright de Sun.

La versión actual de Java es Java SE 6 y se espera que en 2010 aparezca la versión Java SE 7, que presentará el concepto de súper paquetes de código, mejorará la interfaz swing y la creación de documentaciones Javadocs entre otras mejoras.

2.1.3.- Principales características de Java

Orientación a objetos

Java es un lenguaje de programación orientado a objetos. Un objeto es una entidad que combina estado, comportamiento e identidad, es decir, que un objeto es capaz de guardar su estado, tiene información propia, contiene funciones que puede ejecutar y posee un identificador único.

El objetivo principal del paradigma de la programación orientada a objetos, es que la programación sea modular, es decir, que se pueda encapsular dentro de un objeto todo aquello, ya sea información o procedimientos, que actúe sobre el mismo concepto, creando así una entidad única. Pudiendo así crear una representación virtual de un objeto real aunando todas sus propiedades y funcionalidades.

La existencia de objetos, permite que los programas sean más modulares e incluso que un objeto genérico, por ejemplo el objeto libro, sea reutilizado en diferentes programas sin tener que ser modificado, o que se cree el objeto enciclopedia, que herede todas las características del objeto libro, pero que añada sus características y métodos que lo hacen diferente.

Independencia del sistema operativo

Java, al igual que C++, es un lenguaje compilado, lo que significa que el código fuente se compila mediante un compilador diseñado para una plataforma en concreto, generando un archivo ejecutable únicamente por esa plataforma.

Sin embargo Java es independiente de la plataforma que se utilice, esto es porque, además de compilado, Java también es interpretado, lo que permite que, en detrimento de la velocidad, Java pueda ser ejecutado independientemente de la plataforma sobre la que se trabaje.

Este proceso de compilación e interpretación se consigue combinando los conceptos de compilador e intérprete tal y como se describe a continuación.

El código fuente de un programa escrito en Java se compila con un compilador específico para cada plataforma, pero en lugar de generar un archivo ejecutable, este compilador genera un código llamado bytecode, formado por instrucciones específicas de la plataforma Java. Este bytecode se encuentra a medio camino entre el lenguaje Java y el lenguaje máquina. El bytecode se interpreta en tiempo de compilación sobre una máquina virtual de Java (JVM).

La máquina virtual de Java es específica para cada plataforma, lo que permite que un programa Java compilado en cualquier plataforma, sea ejecutado sobre cualquier otra, porque todas las máquinas virtuales entienden el mismo bytecode.

Esta combinación de métodos hace que Java esté a medio camino entre los lenguajes compilados y los interpretados. Es más lento que los compilados, como C++, porque la interpretación en la máquina virtual se realiza en tiempo real, aunque el proceso de compilación en Java sea más rápido que en C++. No obstante es más rápido que los interpretados, como Python, porque gran parte de lo que se analiza en los procesos típicos de interpretación, en Java ha sido ya analizado durante la compilación, lo que reduce las tareas de la máquina virtual.

Esta independencia de plataforma es la que ha logrado que Java tenga un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets o los Java Beans.

Asignación de memoria

Una de las principales diferencias de Java con C++, es que Java es capaz de asignar memoria dinámicamente, con lo que el desarrollador puede olvidarse de tener que pedirle memoria al sistema cada vez que quiera crear un objeto, y de tener que devolvérsela cada vez que borre un objeto.

La diferencia es que en Java la persistencia de los objetos es gestionada directamente por el entorno de ejecución de Java, Java Runtime Environment, JRE en adelante. Una referencia a un objeto es básicamente una dirección de memoria, cuando el JRE detecta que no queda ninguna referencia activa a un objeto, ejecuta un programa interno llamado recolector de basura, garbage collector en inglés, que se encarga de eliminar objetos no referenciados y liberar memoria.

Aunque este proceso de depurado de memoria es automático y gestionado por el JRE, el desarrollador puede hacer una llamada para que el recolector se ejecute, en un cierto punto del código, en segundo plano, es decir, paralelamente a la ejecución del programa.

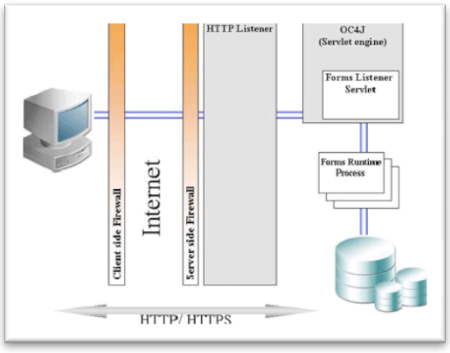
2.1.4.- Usos de Java

Algunos de los usos de la plataforma Java hoy en día son:

- Programación de aplicaciones
- Creación de interfaces gráficas mediante swing
- Comunicación con bases de datos a través de conectores JDBC
- Creación de applets
- Creación de sistemas web mediante servlets y JSP
- Invocación remota de aplicaciones mediante RMI

2.1.5.- Servlets

Un servlet es un objeto, que se ejecuta en un servidor, especialmente diseñado para generar contenido HTML dinámico en respuesta a una petición http generada por un navegador web. En la figura siguiente aparece un diagrama que muestra la arquitectura servlet.



http://www.deakin.edu.au/its/dba/oracledoco/9.0.4.1/9.0.4_doc_library/web.904/b10470/img/ofsarch.gif

Cuando un servlet recibe una petición, el servidor carga y ejecuta el servlet. A continuación el servlet procesa la petición y envía una respuesta, típicamente una web en HTML, pero también puede enviar contenido multimedia en forma de archivo a descargar.

Los servlets son multi-thread, lo que significa que el servidor genera un hilo de ejecución para cada petición recibida, por lo que es posible gestionar varias peticiones al mismo tiempo.

2.2.- SQL

En el proyecto se utiliza un sistema gestor de bases de datos relacionales, en concreto MySQL, que funciona sobre el lenguaje relacional SQL.

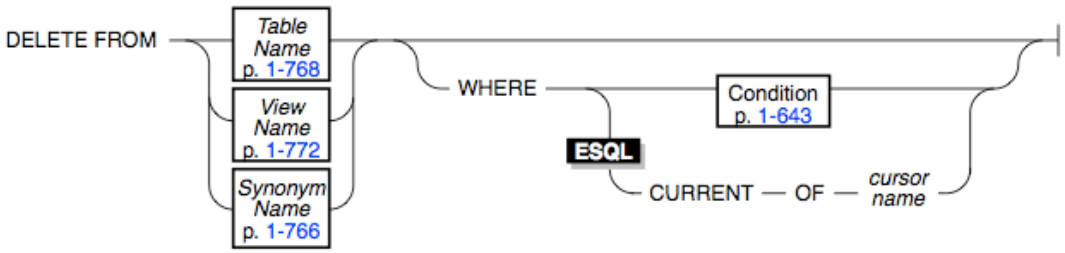
El SQL es un lenguaje estandarizado de definición y manipulación de datos en bases de datos relacionales, por lo que si se utiliza una base de datos relacional, se debe utilizar SQL. A continuación se ofrece una introducción a este lenguaje.

2.2.1.- Introducción

El SQL (Structured Query Language) es un lenguaje utilizado para realizar operaciones de definición, creación y consulta de datos en sistemas gestores de bases de datos relacionales.

El SQL fue estandarizado por el ANSI (American National Standards Institute) y luego por la ISO (International organization for standardization). La versión estándar más reciente es la SQL:2008. Sin embargo no es un estándar de facto, ya que cada uno de los diferentes sistemas gestores de bases de datos, en adelante SGBD, implementa su propia versión del SQL, añadiendo o eliminando funcionalidades, aunque las estructuras básicas se mantienen. Por ejemplo, el estándar SQL define una operación llamada aserción, que consiste en activar una respuesta automática en caso de que se detecte una cierta situación definida por el administrador de la base de datos, sin embargo esta funcionalidad es imposible de implementar sin afectar considerablemente al rendimiento del sistema, por lo que ningún SGBD las implementa.

El SQL está formado por comandos, cláusulas, operadores y operaciones de agregación. La combinación de estos elementos sirve para crear operaciones de consulta, modificación y creación de bases de datos. A continuación se muestra la sintaxis de la operación delete, tal y como se define en el SGBD Informix de IBM.



Las sentencias SQL se dividen en dos grandes tipos:

- DDL (Data Definition Language): Son las sentencias que permiten crear, definir y manipular bases de datos, campos e índices. Por ejemplo las sentencias create y delete.
- DML (Data Manipulation Language): Son las sentencias que permiten realizar consultas y filtrar datos. Por ejemplo las sentencias join y select.

2.3.- SGBD

Un SGBD, o sistema gestor de bases de datos, es necesario para el correcto uso de una base de datos relacional. A continuación se explica en qué consiste un SGBD, su evolución desde que se inventara el primero y sus características principales.

2.3.1- Introducción

Un SGBD es un sistema gestor de bases de datos. Un SGBD está diseñado para controlar la organización, almacenamiento y gestión de gran cantidad de datos y hacerlos accesibles a los usuarios de manera rápida y eficiente. También se ha de encargar de que los datos almacenados no se pierdan, o que una modificación realizada por un usuario no se llegue a grabar y, por tanto, pueda parecer que nunca se ha llegado a ejecutar.

Un SGBD es, hoy en día, una pieza imprescindible dentro del mundo de la informática, pues representan un método de almacenamiento y gestión de la información con una alta seguridad y fiabilidad. Es por ello que existen una gran cantidad de sistemas SGBD en el mercado, ya sean de código abierto, como MySQL o PostgreSQL, o propietarios como Oracle o SQL Server.

El principal objetivo de un SGBD es que el usuario pueda acceder a los datos sin tener que saber como estos están estructurados y/o almacenados. Es decir, que todos los procesos internos del SGBD se ejecuten de manera transparente al usuario, que lo único que este tenga que hacer sea especificar la sentencia SQL que quiera que se ejecute.

Para realizar esta tarea, el SGBD crea estructuras de información, como por ejemplo los índices, que organizan la información en base a algunos de sus valores y reducen enormemente el tiempo de búsqueda, haciendo las consultas más eficientes.

2.3.2- Historia

Los SGBD comenzaron a surgir con el nacimiento de la informática, sin embargo estas primeras versiones no compartían el concepto de las actuales, si no que más bien eran sistemas diseñados específicamente para gestionar aquellos datos que se tenían. Cada organización tenía su propio sistema SGBD que estaba específicamente programado para gestionar sus estructuras de datos.

Cuando los ordenadores comenzaron a crecer en tamaño, esta especialización de los SGBD comenzó a resultar ineficiente, ya que aunque las estructuras de datos fueran diferentes, todas requerían el mismo tipo de atención. Por lo que comenzaron a aparecer los primeros SGBD de propósito general, como el sistema IDS (Integrated Data Store). Lo que condujo a la necesidad de un proceso de estandarización en el mundo de las bases de datos.

El primer modelo de SGBD que apareció, fue el del SGBD navegacional, en el que los datos estaban enlazados entre sí y cada vez que se realizaba una consulta, había que recorrer toda la lista de datos. Un sistema de este tipo fue el SGBD IMS, creado por IBM en 1968.

En 1970 la aparición del modelo relacional, creado por Edgar Codd en 1970, revolucionó completamente el mundo de los SGBD. A raíz de este descubrimiento, en 1973 se inició el proyecto INGRES, en la universidad de Berkeley, dando sus primeros resultados en 1979. A la vez que IBM comenzaba a desarrollar su prototipo de SGBD System R, cuya primera versión apareció en 1975. Aunque no fue hasta 1979 cuando se incorporó un lenguaje estandarizado de búsqueda dentro de un SGBD, el SQL. La eficiencia y funcionalidad de este modelo hicieron que IBM creara un nuevo SGBD basado en el prototipo System R, este ya con propósitos comerciales, el DB2.

Fue a partir de este momento, cuando comenzaron a surgir gran cantidad de proyectos de SGBD como Informix, Sybase, la base del actual Microsoft SQL Server, u Oracle, que surgió basado en el System R de IBM y lo superó en cuota de mercado cuando apareció en 1978.

Desde entonces hasta hoy en día los SGBD han ido creciendo hasta convertirse en lo que son hoy en día, auténticos núcleos de la informática de gestión sin los cuales ya no se concibe ningún tipo de aplicación con un papel relevante dentro de un proyecto.

2.3.3.- Capacidades de un SGBD

Los SGBD de hoy en día normalmente ofrecen las siguientes funcionalidades:

- Consulta y modificación de información.
  Todos los SGBD son capaces de interpretar un lenguaje de consulta y modificación de información, basado en el estándar SQL, que permite a los usuarios especificar los datos que deseen obtener y organizarlos de la manera que les convenga, así como crear nuevos datos o modificar ya existentes.

- Seguridad.
  Un SGBD permite gestionar diferentes tipos de usuarios, cada uno con sus permisos correspondientes, de manera que el administrador de la base de datos pueda controlar que usuarios pueden acceder a que información y cuales pueden modificarla.

- Registro de actividad.
  Un SGBD guarda información de las modificaciones y consultas que han realizado los usuarios en unos registros, de manera que queden constancia de estas y el administrador las pueda consultar en caso de fallo del sistema.

- Gestión de copias de seguridad.
  Un SGBD puede recuperarse de un fallo del sistema o avería eléctrica, mediante la carga de datos de una copia de seguridad y de la información guardada en los registros del SGBD.

- Distribución y replicación de datos.
  La información dentro de un SGBD puede separarse en varios discos y/o localizaciones, con tal de minimizar la carga de los servidores y de reducir el daño que produciría un fallo del sistema o una avería eléctrica.

- Optimización de consultas.
  Los SGBD permiten la creación de una serie de estructuras de información, como los índices, que permiten que, la ejecución de ciertas consultas sean más rápidas y eficientes.

2.3.4.- Gestión de transacciones

Los SGBD trabajan a nivel de transacciones, siendo una transacción una única operación realizada a nivel lógico e independiente de otra transacción.

Las transacciones de un SGBD deben cumplir las siguiente propiedades ACID (Atomicity, Consistency, Isolation, Durability):

- Atomicidad
  Una transacción debe ser atómica, es decir, o no se ejecuta ninguna parte de la transacción, o se ejecuta toda. Si durante la ejecución de la transacción el sistema fallara, el SGBD debe proporcionar un mecanismo para deshacer los cambios realizados por esta. Esto se soluciona mediante el mecanismo de rollback de las bases de datos, que restaura el estado de la base de datos anterior a la ejecución de la transacción.

- Consistencia
  El estado de una base de datos debe mantener su consistencia tanto antes de ejecutar la transacción como después. Si una parte de la transacción introduce información inválida en la base de datos, el SGBD debe desechar la transacción y deshacer todo lo que esta haya modificado.

- Aislamiento
  Una transacción debe ser aislada de todas las demás que se estén ejecutando concurrentemente. El SGBD debe proveer un sistema que evite que una transacción lea datos de la ejecución intermedia de otra transacción. Esto normalmente se implementa mediante un sistema de reservas de la información a la que cada transacción accede.

- Durabilidad
  Una vez que una transacción ha sido ejecutada, sus cambios deben persistir en la base de datos. El SGBD debe suministrar un sistema que evite que se pierdan datos confirmados en caso de fallo eléctrico o avería del sistema. Esto se soluciona guardando información de las modificaciones realizadas en un archivo llamado dietario.

2.4.- HTML

El HTML es un lenguaje estandarizado que se utiliza para transmitir la estructura y el contenido de las páginas web en forma de texto a través de la red.

A continuación se da una breve introducción al HTML, se explican sus componentes básicos y se muestra un pequeño ejemplo de código.

2.4.1.- Introducción

El HTML (HyperText Markup Language) es un lenguaje de marcado para páginas web. Sirve para describir la estructura del texto en un documento, así como para añadir a ese texto otro tipo de elementos como tablas, imágenes, formularios interactivos y otros objetos.

La sintaxis del HTML se basa en el uso de etiquetas o comandos. Estos comandos van encerrados por los símbolos '<' y '>' y la función a la que identifican se aplica a los elementos que están entre la etiqueta de apertura y la de cierre. Por ejemplo <etiqueta>texto</etiqueta>.

Típicamente las etiquetas indican al navegador la forma en la que ha de organizar o presentar al usuario al información encerrada dentro de ellas.

Existe un estándar HTML regulado por el W3C (World Wide Web Consortium), que es implementado por gran parte de los navegadores de Internet. Sin embargo existen navegadores que, como en el caso del SQL, realizan sus propias interpretaciones del HTML, como Internet Explorer. Lo que dificulta la tarea de los diseñadores de páginas web.

2.4.2.- Historia

La primera especificación del HTML apareció en 1991, a manos de Tim Berners Lee, creador del concepto de Internet. La primera versión del HTML era muy simple y comprendía 22 etiquetas diferentes, de las cuales 13 todavía existen en la versión actual de HTML.

Tras un periodo de desarrollo y expansión del HTML, en 1994 se creó un grupo de trabajo, el HTML Working Group, que en 1995 publicó la versión de HTML 2.0 que trataba de ser un estándar sobre el cual se basaran las posibles futuras versiones del HTML.

En este punto la aparición de diferentes versiones del estándar, forzaron la creación de un organismo que se encargara de mantener los diferentes estándares de Internet, la W3C, que ha regulado las diferentes versiones del HTML desde la 2.0.

La versión 3.0, la primera creada enteramente por el W3C, apareció en enero de 1997. Su principal característica es que eliminó los elementos que permitían definir fórmulas matemáticas, producto de las primeras versiones realizadas por el CERN.

El cambio fundamental se produjo con el salto a la versión 4 en 1999, en la que todos los elementos que daban estilo se separaron de la sintaxis HTML y pasaron a incluirse en archivos de estilo en cascada llamados CSS. Lo que otorgó una mayor claridad al código HTML y facilitó en gran medida la aplicación de estilo a los diseños realizados.

La versión actual es la 4, aunque la versión 5 del estándar HTML está en proceso de desarrollo.

2.4.3.- Etiquetas básicas

Existe una serie de etiquetas que deben aparecer en todo documento HTML. Típicamente estas etiquetas definen el tipo de documento HTML y la etiqueta raíz del documento. Otros tipos de etiquetas sirven para crear listas de elementos, tablas o saltos de línea entre otras cosas. A continuación aparecen algunas e:

- <!DOCTYPE>
  Esta etiqueta permite definir el tipo de documento que se usa.
- <HTML>
  Esta es la etiqueta raíz del documento HTML, puede contener atributos que indiquen el lenguaje que se emplea y la codificación del texto.
- <HEAD> <BODY>
  Estas dos etiquetas indican el principio de la cabecera y el cuerpo de la página respectivamente.
- <h1>
  Permite definir el texto que aparezca entre las etiquetas como cabecera de tipo 1
-
  Permite crear una nueva división en la página y asignarle un identificador.
- <table>
  Crea una tabla a la que se pueden añadir filas y celdas
- <form>
  Crea un formulario para que el usuario pueda introducir datos y enviarlos

2.4.4.- Ejemplo

A continuación se muestra un ejemplo de una página web cuyo título es Hello World Document y simplemente muestra el texto Hello, Mundo! en letras grandes.

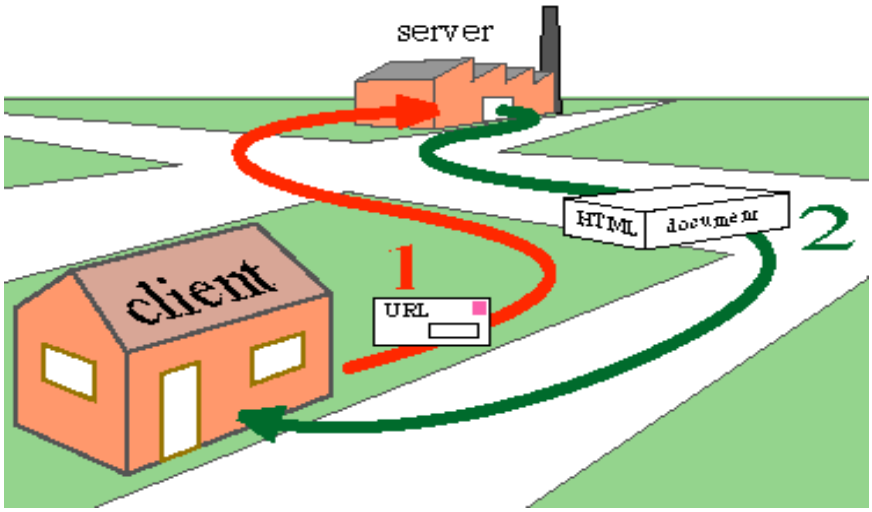
```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
        <head>
                <title>
                        Demostracion de hola mundo
                </title>
        </head>
        <body>
                <h1>
                        Hola, mundo!
                </h1>
         </body>
</html>
```

2.5.- Arquitectura cliente-servidor

En el entorno de las comunicaciones, a través de Internet, entre dos ordenadores, ya sea para acceder a una web, a un servidor de correo o a una base de datos, funcionan baja el paradigma de la arquitectura cliente-servidor.

La comunicación básica funciona de la siguiente manera:

- Un programa cliente, típicamente un navegador de Internet, envía una petición a un servidor.
- Esta petición circula por las diferentes redes que intercomunican el cliente con el servidor, normalmente haciendo uso del protocolo TCP/IP.
- El servidor recibe la petición, la procesa y genera una respuesta adecuada.
- La respuesta se envía al cliente y este la muestra al usuario.



Esquema que muestra la arquitectura cliente-servidor
http://public.web.cern.ch/public/Objects/About/ClientServer.gif

La utilización de esta arquitectura, donde el cliente no almacena ninguna parte de los datos ni de la lógica de programa, frente a otras en las que los clientes poseen alguna parte de estos dos elementos, proporciona diversas ventajas, como por ejemplo:

- Mayor facilidad de mantenimiento. Los servidores pueden replicarse o distribuirse, con tal de realizar tareas de mantenimiento sobre ellos, todo esto sin que el cliente note los cambios.
- Todos los datos están almacenados en los servidores, lo que garantiza un mayor control de la seguridad de los datos, y que solo los clientes autorizados podrán modificar los datos.

En este proyecto la comunicación es vía web, es decir, el cliente envía una petición http al servidor, con los datos necesarios y este le responde enviándo la página web o el archivo solicitado.

El diseño del servidor se ha realizado controlando la autentificación de los usuarios y la seguridad y privacidad de los datos, para que nadie no autorizado pueda acceder a información sensible.

2.6.- Servidores web

Un servidor web es un programa cuyo principal cometido es atender peticiones http de clientes y servir las respuestas indicadas. Las respuestas se sirven codificadas en la respuesta http, lo que permite codificar cualquier tipo de archivo, solamente con indicar su tipo en la respuesta.

Un servidor puede ofrecer contenido estático, en la misma URL con la que se llama al servidor, se indica la dirección del archivo al que se quiere acceder, o contenido dinámico, donde la URL en lugar de apuntar a un archivo en concreto, llama a un programa que es el encargado de generar la respuesta conveniente. Este es el caso de interfaces como servlets, php o asp.

Un servidor mantiene abierta una sesión para cada usuario que se conecta, lo que permite controlar el acceso de los usuarios, ya que la sesión permite guardar atributos como nombres de usuario y niveles de acceso.

Por defecto un servidor atiende peticiones en el puerto 80, ya que este es el puerto asignado para comunicaciones http. No obstante, si se está utilizando una comunicación cifrada, por ejemplo https, el puerto estándar es el 443. Si el servidor se encuentra en un puerto diferente a estos, hay que especificarlo en la URL.

2.6.1.- Jakarta Tomcat

Para la implementación de este proyecto se ha utilizado el servidor web Apache Tomcat. Este servidor es, específicamente, un contenedor de servlets, implementa la especificación de Java Servlets y provee un entorno de trabajo que funciona enteramente sobre Java.

Jakarta Tomcat consta de tres módulos principales: Catalina, que implementa la interfaz de Servlets, Coyote que es el conector http, es el encargado de atender y procesar las peticiones y las respuestas y por último Jasper, que implementa la interfaz de JSP, una interfaz de programación similar a los servlets.

3.- Análisis de datos

En este apartado se detalla el proceso de creación del modelo de datos necesario para el correcto funcionamiento del sistema.

3.1.- Introducción

Una vez decidido el diseño que se iba a seguir para la construcción del sistema, en este caso una plataforma web, lo primero que había que hacer era un análisis del proceso de gestión, con tal de averiguar qué información necesitaba el sistema para funcionar.

El análisis de datos se divide en dos etapas, la primera de estudio del sistema de información requerido y la segunda de elaboración del modelo de datos.

La realización de la primera etapa permitió la identificación de los elementos principales con los que había que trabajar y sus características. Con lo que ya se podía pasar a la segunda etapa.

Dado que en el diseño los datos se almacenan en una base de datos relacional, había que construir un modelo de datos relacional, a partir de los elementos identificados, junto con sus características. Para tal tarea se optó por la elaboración de un diagrama entidad-relación, ya que permite, en un solo diagrama, observar las diferentes entidades, sus atributos y las relaciones que guardan con las demás entidades, es decir, con este tipo de diagramas se puede obtener una idea global del conjunto de entidades del sistema, junto con las conexiones entre ellas.

Para la realización de este proceso de análisis de datos se siguió una metodología iterativa, por lo que cada etapa se ejecutaba y después se estudiaba el resultado, si este no era satisfactorio, se volvía a repetir el proceso, hasta obtener el visto bueno, punto en el cual se pasa a la siguiente etapa en el proceso.

3.2.- Modelo relacional

El objetivo del modelo relacional es la representación de los datos de manera que estos puedan ser introducidos y consultados en una base de datos relacional. El modelo solamente se ocupa de definir los datos de manera que se pueda especificar exactamente que información se tiene y que información se quiere obtener. Es el sistema gestor de bases de datos el que se ocupa de gestionar cómo y dónde se almacenará esta información.

El modelo relacional consiste en la representación de los datos de manera que aquellas entidades de información que estén relacionadas, permanezcan referenciadas por algunos de los atributos que las identifican

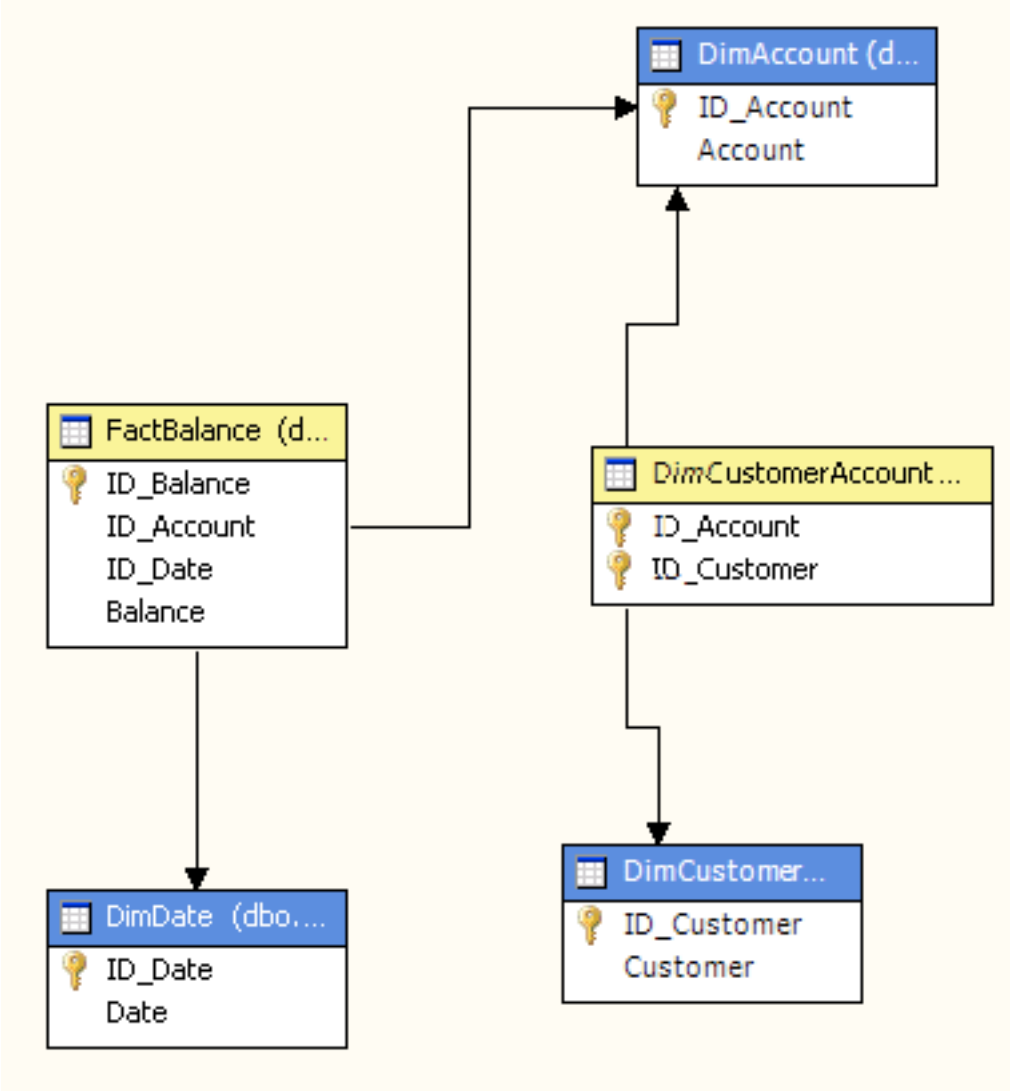
En el modelo relacional, las entidades se identifican por una serie de atributos, llamados clave primaria, estos atributos deben ser únicos, no puede haber dos elementos de la misma entidad con los mismos valores de estos atributos.

Si una entidad A quiere referenciar a otra entidad B, esta entidad A debe almacenar en sus atributos, los valores que forman parte de la clave de B, y pasan a ser una clave

foránea, es decir, que mediante esta clave se pueden acceder a los demás atributos de la entidad B.

Típicamente un diagrama de entidad relación sirve para identificar estas estructuras de datos y las relaciones que entre ellas hay.

A continuación se muestra un ejemplo de diagrama entidad-relación, en el que se pueden observar dos entidades principales, que hacen referencia a otras entidades a las que apuntan con una flecha. La referencia a esta entidades se ha implementado guardando los atributos clave de estas, los marcados con una llave, dentro de los atributos de las entidades principales.



3.3.- Estudio de las entidades principales

La primera etapa del análisis de datos consistió en la obtención de una lista de las entidades principales del sistema, junto con las relaciones que debía haber entre estas.

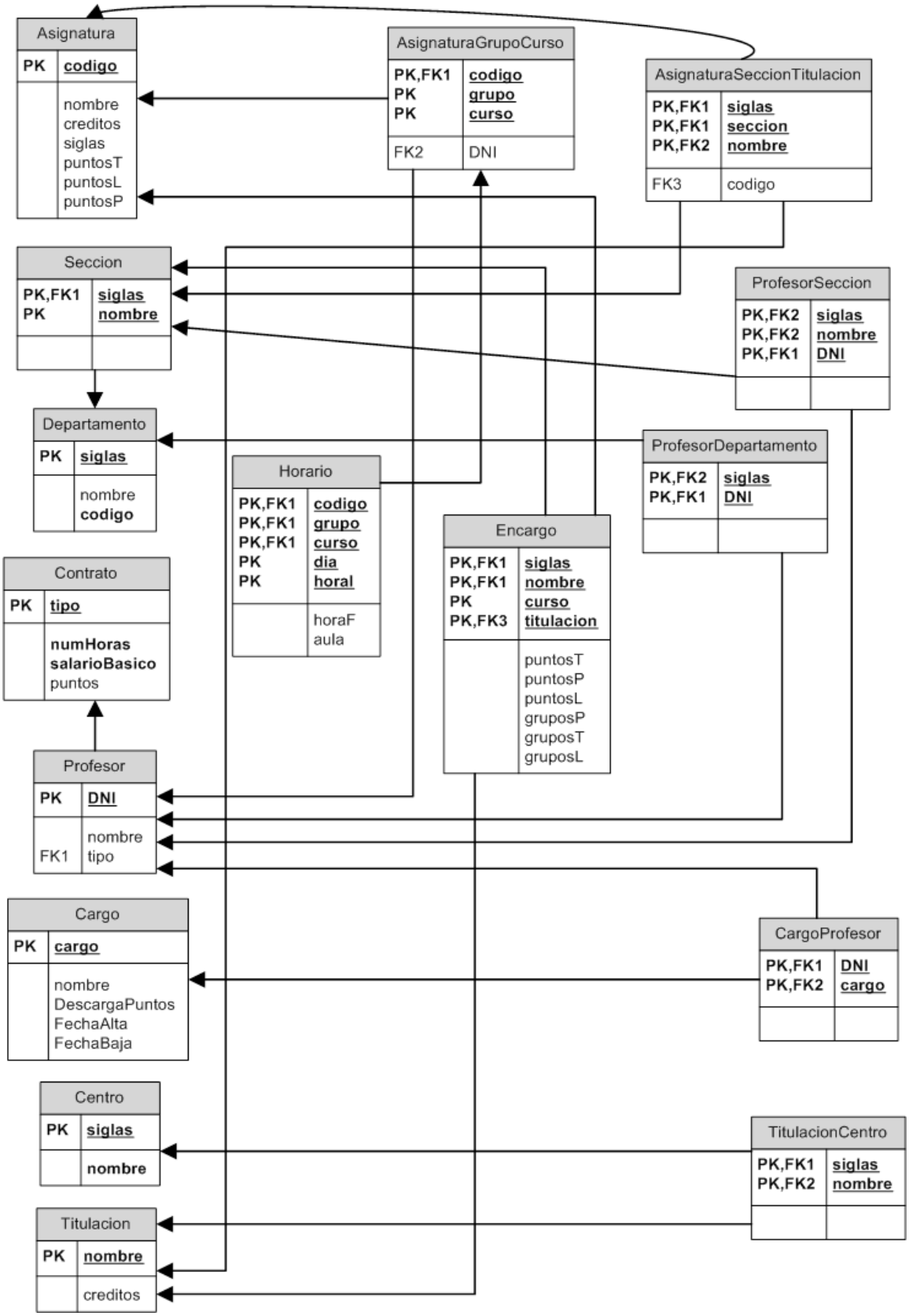
Este proceso produjo como resultado la lista de conceptos que se especifica a continuación.

- **Departamento**
  Se identifica por sus siglas, se debe almacenar su nombre completo. Un departamento está estructurado en varias secciones.
- **Sección**
  Se identifica por sus siglas, se debe almacenar su nombre. Una sección puede está asignada a un centro.
- **Profesor**
  Un profesor se identifica por su DNI. Se deben almacenar su nombre, el tipo de contrato al que está sujeto, su nivel de permisos de acceso y su contraseña para acceder al sistema. Un profesor puede estar asignado a varias secciones y a varios departamentos. Un profesor puede tener uno o más cargos dentro del departamento.
- **Centro**
  Un centro se identifica por sus siglas, además hay que guardar su nombre.
- **Titulación**
  Una titulación se identifica por sus siglas, también hay que guardar su número de créditos totales. Una titulación se puede impartir en diferentes centros.
- **Cargo**
  Un cargo se identifica por su nombre y la fecha de alta del mismo. Se asigna a un único profesor y se almacena información sobre su fecha de baja, si la tiene y la descarga de puntos cuya posesión implica.
- **Contrato**
  Un contrato se identifica por su tipo. Hay que guardar su salario básico y el número de horas estipuladas.
- **Asignatura**
  Una asignatura se identifica por su código. Hay que guardar sus siglas, su nombre y los créditos que vale. Una asignatura se imparte por una sección de un departamento para una titulación. Una asignatura se imparte durante un cuatrimestre y puede haber varios grupos para la misma.
- **Encargo**
  Un encargo es una petición que la universidad a un departamento. Consiste en la definición de los créditos que tiene cada asignatura, tanto de teoría, laboratorio como problemas. En un encargo se indica el número de grupos de teoría, problemas y laboratorio que se han de crear.
- **Horario**
  El sistema debe guardar información sobre los horarios de los profesores, es decir, las horas a las que comienzan y terminan cada una de las horas de clase que imparten y las aulas en las que las impartirán.

Una vez definidas las principales entidades del modelo de datos, se procedió a la creación, a partir de este, del modelo relacional, que es el necesario para trabajar con una base de datos relacional.

El modelo relacional se basa en que las entidades que quieran referenciar a alguna otra del modelo, deben incluir, entre sus atributos, aquellos que identifican a esta. El uso de este modelo garantiza un control sobre las entidades referenciadas, que le otorga robustez al modelo.

A continuación se adjunta el diagrama entidad-relación de este conjunto de datos. Las flechas indican que la tabla origen referencia a la tabla destino.



4.- Implementación

La implementación del proyecto se ha realizado siguiendo el paradigma del diseño en 3 capas: datos, dominio y presentación. La capa de datos gestiona la persistencia de la información manejada, la capa de dominio realiza toda la lógica y la capa de presentación muestra la información al usuario.

A continuación se detalla el proceso de implementación de cada una de las capas.

4.1- CAPA DE DATOS

4.1.1.- Implementación del modelo de datos en MySQL

En este apartado se muestra el proceso de creación de todas las tablas mostradas en el diagrama anterior. Se detallan todos los atributos de cada una y se añade la sentencia SQL de creación de la tabla.

Todas las tablas se han creado sin utilizar la herramienta gráfica MySQL Query Browser, que viene incluida con el SGBD, es decir, se han creado directamente sobre SQL.

Un problema que se encontró a la hora de crear las tablas fue que las claves foráneas, referencias a otras tablas, no funcionaban. Para descubrir que pasaba se recurrió a documentación proveída por el fabricante, donde se indicaba que para utilizar claves foráneas, había que indicar explícitamente los motores de búsqueda y almacenamiento, del SGBD, que había que utilizar, es por ello que todas las sentencias acaben con la frase engine=innodb.

4.1.1.1.- Creación de tablas

Asignatura

Esta tabla contiene información sobre las asignaturas.

Atributo	**Tipo**	**Explicación**
Código	Entero	Clave primaria
Nombre	Cadena de 40 caracteres	Nombre de la asignatura
Siglas	Cadena de 10 caracteres	Siglas de la asignatura
Créditos	Decimal en punto flotante	Créditos totales
PuntosT	Decimal en punto flotante	Créditos de teoría
PuntosP	Decimal en punto flotante	Créditos de problemas
PuntosL	Decimal en punto flotante	Créditos de laboratorio

create table asignatura (
codigo INT unsigned primary key, nombre varchar (40), creditos float, siglas varchar (10), puntosT float, puntosP float, puntosL flota
) engine=innodb;

AsignaturaGrupoCurso

Esta tabla almacena los grupos de cada asignatura que existen en cada curso

Atributo	**Tipo**	**Explicación**
Código	Entero	Clave primaria, clave foránea a asignatura
Grupo	Cadena de 5 caracteres	Clave primaria
Cuatrimestre	Cadena de 5 caracteres	Clave primaria
PuntosTotales	Decimal en punto flotante	Puntos totales del grupo

create table asignaturaGrupoCurso (
codigo int unsigned, grupo varchar (5), cuatrimestre varchar (5),
puntosTotales double, primary key (codigo, grupo, cuatrimestre),
foreign key (codigo) references asignatura (codigo)

) engine=innodb;

AsignaturaGrupoCursoProfesor

Esta tabla almacena las asignaciones de profesores a grupos. Esta información no se podía almacenar en la tabla anterior, porque puede haber más de un profesor en cada grupo y si se usara la tabla anterior se estaría replicando información existente, lo que haría la base de datos más difícil de mantener.

Atributo	**Tipo**	**Explicación**
Código	Entero	Clave primaria, clave foránea a asignatura, clave foránea a AsignaturaGrupoCurso
Grupo	Cadena de 5 caracteres	Clave primaria, clave foránea a AsignaturaGrupoCurso
Cuatrimestre	Cadena de 5 caracteres	Clave primaria, clave foránea a AsignaturaGrupoCurso
Profesor	Entero de 8 posiciones	Clave primaria, clave foránea a profesor
PuntosAsignados	Decimal en punto flotante	Puntos asignados al profesor en este grupo

create table asignaturaGrupoCursoProfesor (

codigo int unsigned, grupo varchar (5), cuatrimestre varchar (5), profesor int (8), puntosAsignados double, primary key (codigo, grupo, cuatrimestre, profesor), foreign key (codigo, grupo, cuatrimestre) references asignaturaGrupoCurso (codigo, grupo, cuatrimestre), foreign key (profesor) references profesor (DNI)

) engine=innodb;

AsignaturaSeccionTitulacion

Esta tabla almacena la correspondencia entre las asignaturas, las secciones que las imparten y las titulaciones en las que se imparten.

Atributo	**Tipo**	**Explicación**
Asignatura	Entero de 8 posiciones	Clave primaria, clave foránea a asignatura
Seccion	Cadena de 15 caracteres	Clave primaria, clave foránea a sección
Titulacion	Cadena de 10 caracteres	Clave primaria, clave foránea a titulación

create table asignaturaSeccionTitulacion (

asignatura int unsigned, seccion varchar (10), titulacion varchar (10), primary key (asignatura, seccion, titulacion), foreign key (asignatura) references asignatura (codigo), foreign key (seccion) references seccion (siglas), foreign key (titulacion) references titulacion (siglas)

)engine=innodb;

Cargo

Esta tabla almacena los diferentes cargos existentes y sus asignaciones a profesores

Atributo	**Tipo**	**Explicación**
Nombre	Cadena de 40 caracteres	Clave primaria
Profesor	Entero de 8 posiciones	Clave primaria, clave foránea a profesor
FechaAlta	Cadena de 10 caracteres	Clave primaria, fecha de alta del cargo
DescargaPuntos	Decimal en coma flotante	Clave primaria

FechaBaja	Cadena de 10 caracteres	Fecha de baja del cargo

create table cargo (

nombre varchar(40), profesor int (8), fechaAlta varchar (10), descargaPuntos flota, fechaBaja varchar (10), primary key (nombre, profesor, fechaAlta), foreign key (profesor) references profesor (DNI),

)engine=innodb;

Centro

Esta tabla guarda información sobre las siglas de un centro y su nombre

Atributo	**Tipo**	**Explicación**
Siglas	Cadena de 10 caracteres	Clave primaria
Nombre	Cadena de 40 caracteres	Nombre del centro

create table centro (

siglas varchar (10), nombre varchar (20), primary key (siglas), foreign key (siglas) references departamento (siglas)

)engine=innodb;

Contrato

Esta tabla almacena información sobre los diferentes tipos de contratos.

Atributo	**Tipo**	**Explicación**
Tipo	Cadena de 20 caracteres	Clave primaria
numHoras	Entero	Número de horas especificadas
salarioBasico	Decimal en coma flotante	Salario básico definido
Puntos	Decimal en coma flotante	Número de puntos lectivos especificados

create table contrato (

tipo varchar(20) primary key, numHoras int unsigned, salarioBasico float, puntos float

)engine=innodb;

Departamento

Esta tabla contiene información básica sobre los diferentes departamentos.

Atributo	**Tipo**	**Explicación**
Siglas	Cadena de 10 caracteres	Clave primaria
Nombre	Cadena de 40 caracteres	Nombre del departamento
Codigo	Entero	Código del departamento

```
create table departamento (
        siglas varchar (10) primary key, nombre varchar (40), codigo int unsigned
) engine=innodb;
```

Encargo

Esta tabla contiene información sobre los encargos de asignaturas que realiza la universidad para una asignatura.

Atributo	**Tipo**	**Explicación**
siglasCentro	Cadena de 10 caracteres	Clave primaria, clave foránea a centro
siglasTitulacion	Cadena de 10 caracteres	Clave primaria, clave foránea a titulación
Cuatrimestre	Cadena de 5 caracteres	Clave primaria, cuatrimestre del encargo
Codigo	Entero	Clave primaria, clave foránea a asignatura
gruposP	Entero	Número de grupos de problemas que tiene el encargo
gruposT	Entero	Número de grupos de teoría que tiene el encargo
gruposL	Entero	Número de grupos de laboratorio que tiene el encargo

create table encargo (
siglasCentro varchar (10) not null, siglasTitulacion varchar (10) not null, cuatrimestre varchar (5) not null , codigo integer unsigned, gruposP int unsigned, gruposT int unsigned, gruposL int unsigned,
primary key (siglasCentro, siglasTitulacion, cuatrimestre, codigo),
foreign key (siglasTitulacion) references titulacion (siglas),
foreign key (siglasCentro) references centro (siglas),
foreign key (codigo) references asignatura (codigo)
) engine=innodb;

Horario

Esta tabla contiene información sobre los horarios de los profesores, en que aula se imparte cada hora de cada grupo que dicho profesor tenga asignado.

Atributo	**Tipo**	**Explicación**
Codigo	Entero	Clave primaria, clave foránea a asignatura
Grupo	Cadena de 5 caracteres	Clave primaria, clave foránea a AsignaturaGrupoCursoProfesor
Cuatrimestre	Cadena de 5 caracteres	Clave primaria, clave foránea a AsignaturaGrupoCursoProfesor
Profesor	Entero de 8 posiciones	Clave primaria, clave foránea a AsignaturaGrupoCursoProfesor
Dia	Cadena de 10 caracteres	Clave primaria, indica el día de la clase.
horaIni	Tiempo (hh:mm:ss)	Clave primaria, indica la hora en la que comienza la clase
horaFin	Tiempo (hh:mm:ss)	Indica la hora en la que finaliza la clase
Aula	Cadena de 15 caracteres	Indica el aula en la que se realiza la clase

create table horario (
codigo int unsigned, grupo varchar (5), cuatrimestre varchar (5), profesor int (8), dia varchar (10), horaIni time, horaFin time, aula varchar(15),
primary key (codigo, grupo, cuatrimestre, profesor, dia, horaIni),
foreign key (codigo, grupo, cuatrimestre, profesor) references asignaturaGrupoCursoProfesor (codigo, grupo, cuatrimestre, profesor)
) engine=innodb;

Profesor

Esta tabla contiene información básica sobre los profesores. Uno de los campos que se almacenan es la contraseña del profesor. Cabe destacar que, por motivos de seguridad,

no se almacena la contraseña, si no un hash de la misma, obtenido por el algoritmo SHA2, explicado con más detalle en el apartado de implementación de login, de manera que resulta imposible obtener la contraseña original a partir del hash almacenado.

Atributo	**Tipo**	**Explicación**
DNI	Entero de 8 posiciones	Clave primaria, DNI del profesor
Nombre	Cadena de 50 caracteres	Nombre completo del profesor
Contrato	Cadena de 20 caracteres	Clave foránea a contrato. Tipo de contrato al que está sujeto
Password	Cadena de 80 caracteres	Hash en SHA2 de la contraseña del profesor
Nivel	Entero	Nivel de permisos de acceso del profesor

create table profesor (
DNI int (8) primary key, nombre varchar (50), contrato varchar (20), password varchar (80), nivel integer, foreign key (contrato) references contrato (tipo)
)engine=innodb;

ProfesorDepartamento

Esta tabla almacena las asignaciones de profesores a departamentos

Atributo	**Tipo**	**Explicación**
DNI	Entero de 8 posiciones	Clave primaria, clave foránea a profesor
siglas	Cadena de 10 caracteres	Clave primaria, clave foránea a departamento

create table profesorDepartamento
(DNI int (8), siglas varchar (10), primary key (DNI, siglas),
foreign key (DNI) references profesor (DNI),
foreign key (siglas) references departamento (siglas)
)engine = innoDB;

ProfesorSeccion

Esta tabla almacena las asignaciones de profesores a secciones

Atributo	Tipo	Explicación
DNI	Entero de 8 posiciones	Clave primaria, clave foránea a profesor
siglas	Cadena de 15 caracteres	Clave primaria, clave foránea a sección

```
create table profesorSeccion (
        DNI int (8), siglas varchar (15), primary key (DNI, siglas),
        foreign key (DNI) references profesor (DNI),
         foreign key (siglas) references seccion (siglas)
)engine = innoDB;
```

Sección

Esta tabla almacena información básica sobre las secciones

Atributo	Tipo	Explicación
siglas	Cadena de 15 caracteres	Clave primaria
Departamento	Cadena de 10 caracteres	Clave foránea a departamento. Departamento de la sección

```
create table seccion (
        siglas varchar (15), departamento varchar (10), primary key (siglas),
        foreign key (departamento) references departamento (siglas)
)engine=innodb;
```

SecciónCentro

Esta tabla almacena la correspondencia entre las secciones y los centros en los que estas imparten asignaturas.

Atributo	Tipo	Explicación
seccion	Cadena de 15 caracteres	Clave primaria, clave foránea a sección
centro	Cadena de 10 caracteres	Clave primaria, clave foránea a centro.

```
create table seccionCentro (
        seccion varchar (10), centro varchar (10), primary key (seccion, centro),
        foreign key (seccion) references seccion (siglas),
        foreign key (centro) references centro (siglas)
)engine=innodb;
```

Titulación

Esta tabla contiene información básica sobre las titulaciones.

Atributo	**Tipo**	**Explicación**
siglas	Cadena de 10 caracteres	Clave primaria
nombre	Cadena de 40 caracteres	Nombre de la titulación
Creditos	Decimal en coma flotante	Número de créditos de la titulación

```
create table titulacion (
        siglas varchar (10) primary key, nombre varchar (40), creditos float
) engine=innodb;
```

TitulaciónCentro

Esta tabla almacena la correspondencia entre las titulaciones existentes y los centros en las que se imparten.

Atributo	**Tipo**	**Explicación**
siglasCentro	Cadena de 10 caracteres	Clave primaria, clave foránea a centro
siglasTitulacion	Cadena de 10 caracteres	Clave primaria, clave foránea a titulación

```
create table titulacionCentro (
        siglasCentro varchar (10), siglasTitulacion varchar (10),
        primary key (siglasCentro, siglasTitulacion),
        foreign key (siglasCentro) references centro (siglas),
        foreign key (siglasTitulacion) references titulacion (siglas)
)engine = innoDB;
```

4.1.2.- Comunicación entre MySQL y el servlet

La comunicación entre MySQL y el servlet creado se realiza a través del conector JDBC que Java dispone para la conexión con MySQL.

Se ha implementado una clase que gestiona y centraliza todas las conexiones con la base de datos. De manera que cualquiera de las otras clases que deban conectarse a la base de datos, solamente tienen que llamar a las operaciones de esta clase para hacerlo. Todas las funciones y variables de esta clase se han implementado de manera estática, es decir, no son parte de un objeto residente en memoria, que se crea y se destruye cada vez, si no que su posición en memoria es fija, lo que hace que cada vez que se acceda a estas funciones, no se tenga que preguntar donde está, ya que el gestor de memoria de Java ya conoce su posición, fijada en el proceso de compilación, lo que aumenta el rendimiento del sistema.

```
import java.sql.*;

public class bd{

        private static String driver=" ";
        private static String ruta=" ";
        private static String username=" ";
        private static String password=" ";

        public static Connection conectar ()throws Exception{
            Class.forName(driver);
            Connection conexion = DriverManager.getConnection (ruta,username, password);
            return conexion;
        }

        public static void desconectar (Connection con)throws Exception{
            con.close();
        }

        public static ResultSet consulta (Connection con, String query)throws Exception{
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery (query);
            return rs;
        }

        public static int actualizacion (Connection con, String query)throws Exception{
            Statement st = con.createStatement();
            int num = st.executeUpdate (query);
            return num;
        }
}
```

4.1.2.1.- Configuración del acceso a la base de datos

Como se puede observar en el código adjuntado en el apartado anterior, se han creado una serie de variables que contendrán parámetros como la ruta del driver JDBC

necesario para conectar con la base de datos y la ruta, el nombre de usuario y la contraseña de la base de datos. Estas variables se crean vacías, ya que su contenido se obtendrá en tiempo de ejecución, al arrancar el servidor web, leyendo un archivo de configuración que proveerá la información necesaria para la conexión.

4.1.2.2.- Operaciones de la clase

- Conectar
  La conexión con la base de datos se realiza a través de la función conectar. Esta función no recibe ningún parámetro, pues lo que necesita para realizar la conexión se ha indicado en tiempo de compilación.
  El primer paso es obtener el conector JDBC para MySQL, lo cual se hace pidiéndolo a Java el driver adecuado, pasándole el nombre del mismo.
  Una vez se tiene el driver, hay que establecer la conexión, para lo que se necesita el nombre de usuario y la contraseña, que tengan los permisos necesarios en la base de datos, además de la dirección donde se encuentre la base de datos.
  Una vez establecida la conexión se la envía al usuario.

- Desconectar
  Dada la conexión iniciada por el usuario, se cierra la misma terminando así la comunicación con la base de datos.

- Consulta
  La función se llama con dos parámetros, la conexión a la base de datos y un string que contenga la consulta a ejecutar.
  Esta función simplemente ejecuta la consulta sobre la base de datos y devuelve su resultado, en forma de ResultSet, que es la interfaz de JDBC para mostrar resultados de consultas a bases de datos, a quien haya llamado a la función.

- Actualización
  El funcionamiento es igual al de la consulta, la diferencia es que el resultado de esta función es el número de filas de la base de datos actualizadas por la consulta.

4.1.2.3.- Excepciones y errores mostrados

El correcto funcionamiento de esta clase depende del estado del SGBD MySQL y la correctitud de las sentencias ejecutadas.

Si la base de datos no está disponible, se lanza una excepción que indica que no se ha conseguido contactar con el servidor.

Si alguna consulta o actualización contienen errores sintácticos o caracteres prohibidos, se devuelve una excepción que indica el problema.

Si se intenta hacer una actualización con la función de consulta, o viceversa, se lanza una excepción previniendo al usuario.

Por último si una actualización no se puede ejecutar por algún problema de correspondencia con el modelo de datos existente, se avisa al usuario mediante una excepción.

4.2.- CAPA DE DOMINIO

4.2.1.- Clase principal

La clase principal del proyecto es la que se encuentra en el archivo principal.java

Esta clase hereda directamente todas las propiedades de la interfaz servlet de java, por lo que es capaz de comunicarse directamente con la petición http recibida y con la respuesta generada por el servidor Tomcat.

Esta clase consta de tres principales métodos Init, doGet y doPost. A continuación se detalla cada uno de ellos.

El método INIT es llamado directamente por el servidor cuando este arranca, init hace que el servidor cargue el servlet y que este esté disponible para atender peticiones. Es aquí donde se carga el archivo de configuración que define la ruta y el puerto de acceso a la web, así como la información de la conexión a la base de datos, tal y como se puede observar en el siguiente fragmento.

```
public void init (ServletConfig config)throws ServletException {
        super.init(config);
        //Definiciones de variables y apertura de ficheros
        try{
                InputStream is= getServletContext().getResourceAsStream("/WEB-INF/config.txt");
                InputStreamReader isr=new InputStreamReader(is);
                BufferedReader lector= new BufferedReader(isr);
                String ip_server=lector.readLine();
                String puerto_web=lector.readLine();
                String puerto_https=lector.readLine();
                String ruta=lector.readLine();
                bd.driver=lector.readLine();
                bd.ruta=lector.readLine();
                bd.username=lector.readLine();
                bd.password=lector.readLine();
                auxiliares.ip_server=ip_server;
                auxiliares.puerto_web=puerto_web;
                auxiliares.ruta=ruta;
        }
        catch(Exception e){
                        e.printStackTrace();
        }
}
```

La primera línea de este fragmento es la que carga la configuración del servidor e inicia el servlet con esa configuración.

Dentro del bloque try-catch, lo que se hace es, primero leer el archivo de configuración del servlet, config.txt, para ello hay que pedir al servidor la dirección base de su sistema de archivos mediante la instrucción getContext.

Una vez obtenido el archivo, cada una de sus líneas es una variable diferente, por lo que estas líneas se leen en orden y se van asignado los valores leídos a las variables adecuadas.

El método doGet es llamado cuando la petición recibida no contiene ningún tipo de dato pasado por POST, los datos pasados por POST son aquellos que no se codifican como parte de la URL, típicamente son datos que el usuario ha introducido mediante formularios de la web.

Lo primero que hace este código es obtener la sesión abierta en el navegador, mediante la cual se pueden recoger parámetros que sirven para identificar al usuario, como su nombre de usuario, que sirven para comprobar si el usuario que ha realizado la petición había iniciado sesión en la web. Además también se recogen los parámetros pasados por POST en la URL que sirven para identificar que página desea consultar el usuario.

```
HttpSession s = req.getSession(true);
String username = (String)s.getAttribute(s.getId());
String seccion= req.getParameter("sec");
```

Una vez recogidos estos parámetros son analizados y, en función de su valor, se redirige al usuario hacia una clase u otra, como se puede apreciar en el siguiente fragmento de código.

```
if(username==null){
        res.setContentType("text/html");
        PrintWriter out =res.getWriter();
        menuPrincipal menu = new menuPrincipal();
        menu.paginaMenuPrincipal(out, username);
}
```

Este fragmento se ejecuta para comprobar que el usuario se haya registrado en el sistema, en caso contrario, cuando entra en el if, se establece el tipo de respuesta como texto HTML, se obtiene el canal de escritura del servidor, se llama a la función que crea la página de inicio, pasándole el canal de escritura y el nombre de usuario, en este caso null. Este mismo proceso se realiza, con la variable seccion, para comprobar que ha solicitado ver el usuario.

El método doPost realiza prácticamente la misma función que doGet, con la diferencia de que doPost se llama cuando el usuario ha pasado datos por POST, lo que normalmente indica que se ha introducido información mediante formularios.

En el siguiente extracto de código, de la función doPost, se puede apreciar que resulta muy similar a doGet, con la diferencia de que, en este caso, se leen dos variables para identificar que quiere hacer el usuario, una para identificar que está en el menú de centros y otra para identificar la operación quiere ejecutar, consulta, modificación, creación o eliminación.

```
//centro
                else if(seccion.contentEquals("5")){
                        String accion= req.getParameter("acc");
                        centro c = new centro();
                        if(accion !=null){
                                if (accion.contentEquals("1")){

                                                        c.consulta(req, out);
                                }
                                else if(accion.contentEquals("2")){
                                        c.inserta(req, out);
                                }
                                else if(accion.contentEquals("3")){
                                        c.actualiza(req, out);
                                }
                                else if(accion.contentEquals("4")){
                                        c.elimina(req, out);
                                }
                        }
                        else{
                                c.consulta(req, out);
                        }
                }
```

4.2.2.- Clases de presentación de la información

La estructura de clases del servlet se ha presentado en el apartado de introducción de la solución.

Tal y como allí se expresaba, se ha creado una clase para cada uno de los diferentes apartados de la web. Esto se ha hecho así por claridad del código, cada una de estas clases posee los mismos métodos. A continuación se explica cada uno de estos métodos, poniendo como ejemplo los de la clase profesor:

4.2.2.1.- Función consulta

La función consulta muestra la página principal de la sección en la que el usuario se encuentra. Esta función se comunica con la base de datos y muestra al usuario la información correspondiente.

La función está estructurada por partes, en la que cada una va completando una parte de la vista hasta que se termina de generar todo el contenido y se envía la página al controlador de la capa de presentación para que genere la página web entera que verá el usuario.

A continuación se muestra la primera parte de la función consulta

```
Vector contenido= new Vector(0,1);
anadirFiltro(contenido);
//Accedemos a la session
HttpSession s = req.getSession(true);
String nombreConsulta= req.getParameter("nombreConsulta");
String DNIConsulta= req.getParameter("DNIConsulta");
String contratoConsulta= req.getParameter("contratoConsulta");
String departamentoConsulta= req.getParameter("departamentoConsulta");
String seccionConsulta= req.getParameter("seccionConsulta");
```

Esta primera parte crea el vector contenido, que será donde se irá almacenando el código HTML generado para pasarlo al controlador HTMLPrinter que es el que generará la página web entera.
Acto seguido se llama a la función auxiliar anadirFiltro, esta función genera el código HTML que crea los formularios HTML que permitirán al usuario filtrar los profesores que desee, según diferentes campos.
Tras este paso se recoge la sesión del navegador, que permitirá acceder a información del usuario, como su nombre de usuario y su nivel de permisos de acceso.
Por último en esta primera parte se recogen los valores introducidos por el usuario en el formulario del filtro, para generar la consulta SQL adecuada a los requisitos del usuario.

La segunda parte de la función tiene que ver con la generación de la consulta SQL con los parámetros elegidos por el usuario.

```
String set="";
String set2="";
if((nombreConsulta!=null) && !(nombreConsulta.contentEquals(""))){
        set=set + "and p.nombre like '%"+nombreConsulta+"%' ";
}
.
.
if((seccionConsulta!=null) && !(seccionConsulta.contentEquals(""))){
        set2= set2 +"and s.siglas='"+seccionConsulta+"' ";
        dept=true;
}

String query=("select p.DNI, p.nombre, p.contrato, pd.siglas, ps.seccion from profesor p,
profesorSeccion ps, seccion s, profesorDepartamento pd where p.DNI=pd.DNI and
p.DNI=ps.profesor and ps.seccion=s.siglas and pd.siglas=s.departamento "+set+" "+set2+
" order by p.nombre, s.departamento, s.siglas");

String query2=("select p.DNI, p.nombre, p.contrato, d.siglas from profesor p left join
profesorDepartamento d on (p.DNI=d.DNI) where p.DNI not in (select p2.DNI from
profesor p2, profesorSeccion ps, seccion s, profesorDepartamento pd where
p2.DNI=pd.DNI and p2.DNI=ps.profesor and ps.seccion=s.siglas and
pd.siglas=s.departamento ) "+set+" order by p.nombre");
```

Como se puede observar en el anterior fragmento de código, lo que se hace es recoger los parámetros introducidos por el usuario y encajarlos dentro de la sentencia SQL preparada para la consulta. Si el usuario no ha elegido ningún filtro, se ejecuta la sentencia sin aplicar ningún filtro.

En el caso particular de los profesores, hay que realizar dos consultas por separado, la primera, muestra los profesores con sus asignaciones a secciones, mientras que la segunda muestra los profesores que no están asignados a ninguna sección.
Esta es una de las consultas más costosas, ya que en realidad se están ejecutando tres consultas, porque la consulta que busca los profesores sin departamentos ni secciones, debe buscar los profesores que tienen secciones o departamentos asignados y eliminarlos de toda la lista de profesores, por lo que ejecuta dos consultas dentro de una.
Hay que destacar que en el momento que el usuario filtra por sección o departamento, existe una variable booleana que impide la ejecución de esta consulta, lo que reduce el número de consultas reales de tres a una, reduciendo el coste total del proceso.

La tercera parte consiste en la ejecución de la consulta y la inserción de los datos obtenidos en la página que se mostrará al usuario. Como se observa en el siguiente fragmento de código.

```
try{
        Connection con = bd.conectar();
        ResultSet rs = bd.consulta (con, query );
        contenido.addElement("<div id=\"elementos\">");
        contenido.addElement("<table>");
                Vector titulos = new Vector(0,1);
                titulos.addElement("DNI");
                titulos.addElement("Nombre");
                titulos.addElement("Tipo de contrato");
                titulos.addElement("Departamento");
                titulos.addElement("Sección");
                auxiliares.cabecera(titulos,contenido);
                int i=auxiliares.rellenar(contenido, rs, 5, 0);
                if(!dept){
                        rs = bd.consulta (con, query2 );
                }
                i=auxiliares.rellenar(contenido, rs, 4, i, 2);
        contenido.addElement("</table>");
        contenido.addElement("</div>");
        bd.desconectar(con);
        }
catch (Exception e){
        e.printStackTrace();
}
```

Todo el anterior fragmento de código está encapsulado en un bloque try-catch, esta es la forma que tiene Java de encapsular un código que puede lanzar excepciones si falla o se encuentra algún problema. Esta encapsulación se realiza para que el programador pueda controlar qué hacer en caso de fallo del código. En este caso el código problemático es la consulta a la base de datos, pues puede que la no se consiga establecer una conexión con la base de datos, que la consulta esté mal formulada u otro tipo de problemas, que no deberían bloquear el resto de funcionalidades, por lo que se este fragmento se aísla del resto de código de esta manera.
La primera acción que se realiza es la conexión con la base de datos, una vez establecida esta conexión, se realiza la consulta SQL y se crean dos sentencias HTML, , que define una sección de la página web y <table>, que crea una tabla

HTML. A continuación se crean los elementos de la cabecera de la tabla que, mediante la función cabecera, se incorporan al código HTML.
Una vez creada la cabecera, mediante la función rellenar, pasando el resultado de la consulta SQL, se rellena la tabla con la información obtenida de la base de datos.
Tras rellenar la tabla HTML, se cierran las correspondientes etiquetas HTML y DIV y se cierra la conexión con la base de datos, para no dejar una conexión abierta que consumiría recursos y podría ser fuente de problemas, tanto para el servidor web como para el de bases de datos.
En el fragmento catch, que se ejecuta cuando el código lanzara una excepción, lo que se hace es imprimir en el log del servidor el error encontrado por la base de datos, para que el administrador de sistemas tenga constancia del mismo.

La última parte de esta función consiste en la creación de los enlaces a los menús y la llamada al controlador HTMLPrinter.

```
if((String)s.getAttribute(s.getId())!=null){
        anadirPassword(contenido);
}
if(auxiliares.permiso1(s)){
        contenido.addElement("<A
        HREF=\"http://localhost:8080/examples/servlets/servlet/principal?sec=20&o
        pc=1\" onClick=\"return popup(this, 'notes')\">alta</A>");
        contenido.addElement("<A
        HREF=\"http://localhost:8080/examples/servlets/servlet/principal?sec=20&o
        pc=2\" onClick=\"return popup(this, 'notes')\">modificación</A>");
        contenido.addElement("<A
        HREF=\"http://localhost:8080/examples/servlets/servlet/principal?sec=20&o
        pc=3\" onClick=\"return popup(this, 'notes')\">baja</A>");
}
HTMLPrinter html = new HTMLPrinter();
html.printHTML(out, (String)s.getAttribute("nombre"), contenido, true);
}
```

En esta última parte, se comprueba si el usuario tiene un nivel de permisos que le permita realizar altas, bajas o modificaciones de profesores, en caso afirmativo, se incluyen los enlaces que permiten abrir ventanas pop-up con los formularios correspondientes para realizar altas, bajas o modificaciones de profesores.
En último lugar se llama al controlador HTMLPrinter y se le pasa el vector en que se han ido almacenando todos los fragmentos HTML que se han generado durante la ejecución de la función.

4.2.2.2.- Funciones inserta, actualiza y elimina

Las funciones inserta, actualiza y elimina, simplemente recogen la información de los formularios web que las han llamado y se conectan a la base de datos para realizar la operación pertinente. A continuación se muestra un ejemplo de la operación elimina de la clase profesor

```
//Accedemos a la sesion
HttpSession s = req.getSession(true);
if((String)s.getAttribute(s.getId())!=null){
        String DNIDelete= req.getParameter("DNIDelete");
        if (!(DNIDelete.contentEquals("")) && auxiliares.permiso1(s)){
                //Aqui controlariamos los permisos
                try
                {
                        Connection con = bd.conectar();
                        int num2 = bd.actualizacion (con, "delete from
                        profesorSeccion where profesor='"+DNIDelete+"' ");
                        int num3 = bd.actualizacion (con, "delete from
                        profesorDepartamento where DNI='"+DNIDelete+"' ");
                        int num = bd.actualizacion (con, "delete from profesor
                        where DNI='"+DNIDelete+"'");
                        bd.desconectar(con);
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
}
popupBaja(req, out);
```

En esta función, se observa que lo que se ejecuta es, primero una comprobación de los permisos del usuario, con la función auxiliares.permiso1, una vez comprobados estos permisos, se pasa a realizar la conexión a la base de datos y ejecutar las operaciones necesarias sobre la misma. Por último, se llama a la función popupBaja, que es la que ha creado el popup con el formulario de baja, para que el usuario, si lo desea, pueda continuar dando de baja a más profesores, con el formulario actualizado, es decir, tras haber borrado al profesor seleccionado.

4.2.2.3.- Funciones de pop-up

Las funciones de pop-up se encargan de crear formularios para las funciones de alta, baja y modificación en la base de datos, así como para la subida y descarga de archivos en formato xls. Las funciones se llaman popupAlta, popupBaja, popupModifcacion y popupExcel. A continuación se muestra un extracto de código de la función popupAlta de la clase profesor.

```
Vector contenido = new Vector();
contenido.addElement("<div id=\"input2\">");
        contenido.addElement("<form
        action=\"http://"+ip_server+":8080/examples/servlets/servlet/principal?sec=
        3&acc=2\" method=\"POST\">");
...
contenido.addElement("<p>DNI: </p>");

contenido.addElement("<input type=\"text\" name=\"DNIInput\" id=\"DNIInput\"
value=\"\" size=\"8\">");
contenido.addElement("<p>Contraseña: </p>");
...
contenido.addElement("<input type=\"password\" name=\"passwordPInput\"
id=\"passwordPInput\" value=\"\" size=\"10\">");

contenido.addElement("<p>Contrato: </p>");
contenido.addElement("<select type=\"text\" name=\"contratoInput\" >");
auxiliares.seleccion(contenido, "select c.tipo from contrato c");
contenido.addElement("</select>");

contenido.addElement("<input type=\"submit\" value=\"Input\" id=\"botonInput\">");
contenido.addElement("</form>");
contenido.addElement("</div>");
contenido.addElement("<A
HREF=\"http://"+ip_server+":8080/examples/servlets/servlet/principal?sec=3\"
onClick=\"return targetopener(this, 1)\">Volver</A>");

HTMLPrinter html=new HTMLPrinter();
html.printpopup(out, contenido);
```

En el anterior extracto de código se puede observar el proceso de creación de un formulario web. El primer paso que se realiza es crear la etiqueta form con un atributo action, que es la URL de la página que realiza la acción del formulario, en este caso este enlace llevaría a la función inserta de la clase profesor.

Dentro del formulario se crean los diferentes campos en los que el usuario introducirá la información. Hay campos de tipo texto, para que el usuario introduzca una cadena de caracteres, otros campos son de tipo contraseña, que básicamente son iguales que los de texto, pero en los que los caracteres escritos aparecen en la pantalla en forma de puntos, y por último están los de tipo select, que aparecen como un menú desplegable en el que el usuario elige una de las opciones definidas en el código de la página, en este caso se realiza una consulta a la base de datos para obtener los valores existentes.

Una vez creado el formulario, se llama al controlador HTMLPrinter, pero en este caso se llama a la función printpopup, porque lo que se va a mostrar en pantalla es un popup y el código es ligeramente diferente al de una página web normal.

4.2.2.4.- Funciones auxiliares

Durante la ejecución del código se emplean varias funciones auxiliares. La mayoría de las cuales sirven para recorrer el objeto, que devuelve la ejecución de una consulta SQL e integrar los resultados obtenidos con el código HTML de la página, por ejemplo la función rellenar, que rellena las tablas de resultados generadas en la función consulta y la función selección, que genera las opciones para los desplegables de los formularios.
Aparte de estas, hay otras funciones auxiliares que se usan solamente en algunas clases, como la función hashPassword, presentada a continuación.

```
public static String hashPassword(String password) {
        String hash= null;
        try {
        MessageDigest md = MessageDigest.getInstance("SHA");
        md.update(password.getBytes());
        byte raw[] = md.digest(); //step 4
  hash = (new BASE64Encoder()).encode(raw); //step 5

        } catch (NoSuchAlgorithmException nsae) {

        }
  return hash;
}
```

Esta función se utiliza para asegurar la privacidad del almacenamiento de contraseñas en la base de datos. Por motivos de seguridad una contraseña no se puede almacenar en la base de datos, ya que de esta manera, se perdería la privacidad que el uso de la misma garantiza, pues podría ser leída por cualquiera que accediera a la base de datos.
Es por esto que en lugar de almacenar la contraseña, se almacena un hash criptográfico de la misma. Un hash es una función determinista que toma un bloque arbitrario de caracteres y genera un patrón de caracteres mediante un procedimiento matemático unidireccional, de manera que dada una palabra, su hash siempre es el mismo, pero dado un hash es imposible recuperar la palabra original.

También existe otra función auxiliar, llamada comprobar, que se encarga de depurar los valores introducidos por el usuario, para que no haya ningún carácter no permitido.

```
public static String comprobar(String str){

        if(str.matches(".*'*")){
                str="";
        }
        return str;
}
```

Lo que hace esta función es revisar las cadenas de caracteres introducidas por el usuario, mediante el teclado, en busca del símbolo ‘. Esta búsqueda se realiza por motivos de seguridad, detallados a continuación.
Existe un tipo de ataque informático llamado SQL Injection, que consiste en que en lugar de introducir el valor pedido por el formulario, el usuario introduce una sentencia SQL, que, si no se controla por software puede acceder a información no autorizada o borrar tablas de la base de datos. Para realizar dicho ataque, el usuario debe marcar el final de la sentencia, que normalmente se ejecutaría, con el símbolo ‘ por lo que si se detecta dicho símbolo, que no tiene sentido en el contexto de datos que en este proyecto se maneja, se puede descartar la cadena de caracteres recibida, evitando de esta manera un posible acceso no deseado a la base de datos.

Las últimas de las funciones auxiliares que se emplean, son las funciones permiso1 y permiso2, que sirven para controlar el nivel de permisos del usuario conectado a la web. A continuación se muestra como ejemplo la función permiso1.

```
public static boolean permiso1(HttpSession s){
        int nivel=(Integer)s.getAttribute("nivel");
        if(nivel<2)
                return true;
        else
                return false;
}
```

Esta función recibe como parámetro la sesión del usuario conectado en el servidor y consulta el nivel de permisos, que ha sido inicializado en el momento del login del usuario, si el nivel es el correcto, la función devuelve cierto, lo que daría permiso para ejecutar la operación.

4.2.3.- Clase de login y logout

La función de login es una parte esencial de la aplicación, ya que el uso de un login garantiza que usuarios no registrados en el sistema no podrán acceder al mismo, por lo que una buena gestión de los permisos de usuario y niveles de acceso se hace obligatoria, prácticamente para cualquier sistema de este tipo.
A continuación se muestran extractos de la función de login implementada en este caso.

```
//Accedemos a la sesion
        HttpSession s = req.getSession(true);
        String username= req.getParameter("username");
        String password= auxiliares.hashPassword(req.getParameter("password"));
        String valor="";
        String nombre="";
        int nivel=0;
        try
        {
                Connection con= bd.conectar();
                ResultSet rs = bd.consulta(con, "select p.password, p.nombre, p.nivel
from profesor p where p.dni='"+username+"'");
                while (rs.next())
                {
                        valor=rs.getString (1);
                        nombre=rs.getString(2);
                        nivel=Integer.valueOf(rs.getString(3));
                }
                bd.desconectar(con);

        }
```

Como se puede observar, lo primero que se hace es obtener la sesión del servidor, para poder almacenar los datos relativos al usuario. Una vez que se tiene la sesión, se recupera la información introducida por el usuario en el formulario de login, esto es, nombre de usuario y contraseña.

Tal y como se ha explicado anteriormente, en la base de datos se guarda un hash de la contraseña, por lo que a la contraseña introducida por el usuario se le aplica la misma función de hash, con tal de poder compararla con la almacenada en la base de datos.

Acto seguido se accede a la base de datos y se obtienen el hash almacenado y el nivel de accesos para ese nombre de usuario.

```
Vector print_user = new Vector(0,1);
        //guardamos el username en la sesion
        if(password.contentEquals(valor)){
                s.setAttribute(s.getId(), username);
                s.setAttribute("nivel", nivel);
                s.setAttribute("nombre", nombre);
                print_user.addElement("<div id=\"mensaje\">");
                print_user.addElement("<p> Hola "+nombre+"</p>");
                print_user.addElement("</div>");

        }
        else{
                print_user.addElement("<div id=\"mensaje\">");
                print_user.addElement("<p> Login incorrecto</p>");
                print_user.addElement("</div>");
        }
        HTMLPrinter html = new HTMLPrinter();
        html.printHTML(out, (String)s.getAttribute("nombre"), print_user, true);
```

Una vez obtenida la información de la base de datos, se compara el hash almacenado con el calculado a partir de la contraseña proporcionada por el usuario.

En caso afirmativo se estable el nombre de usuario, el DNI, como identificador de la sesión, se añaden el nombre del usuario y su nivel de acceso como atributos de la sesión y se muestra un mensaje de bienvenida.

En caso contrario se muestra un mensaje indicando que el procedimiento de login ha fallado.

En el caso de que el usuario decidiera cerrar la sesión abierta, el código a ejecutar sería el siguiente.

```
// Acceso a la sesion
   HttpSession s = req.getSession(true);
   //Quitamos el usuario de la sesion
   s.removeAttribute(s.getId());
   s.removeAttribute("nivel");
   s.removeAttribute("nombre");
//Mostramos la pagina inicial de la web
   menuPrincipal menu = new menuPrincipal();
   menu.paginaMenuPrincipal(out, null);
```

Este código accede a la sesión del usuario en el servidor y borra el identificador y los demás parámetros establecidos anteriormente, lo que, a efectos prácticos, es como borrar la sesión.

4.2.4.- Clases de subida de archivos

Una parte fundamental de este proyecto es la posibilidad de poder subir archivos en formato xls al servidor, para poder cargar configuraciones.

Es por ello que la parte de la subida de archivos ha sido una de las más estudiadas y cautelosamente diseñadas.

Hay varias clases que se encargan de la subida de archivos, una para cada esquema de archivo que había que tratar. Pero todas estas clases comparte la misma estructura, una función que se encarga de la subida del archivo al servidor y otra que se encarga de procesar el archivo y ejecutar los cambios necesarios en la base de datos. A continuación se presentan extractos de código que ilustran estas funciones.

```
boolean isMultipart = ServletFileUpload.isMultipartContent(req);
if(isMultipart){
        FileItemFactory factory = new DiskFileItemFactory();
        ServletFileUpload upload = new ServletFileUpload(factory);
        List items = upload.parseRequest(req);
        Iterator iter = items.iterator();
        while (iter.hasNext())
        {
                FileItem item = (FileItem) iter.next();
                if (!item.isFormField()){
                        fileName = item.getName();
                        sizeInBytes = item.getSize();
                        DateFormat dateFormat = new SimpleDateFormat("yyyy:MM:dd-HH:mm:ss");
                        Date date = new Date();
                        File uploadedFile = new File(dateFormat.format(date)+".xls");
                        item.write(uploadedFile);
                        tratarExcel(req, out, contenido, uploadedFile);
                }
        }
contenido.addElement("<br><h1>success ....."+fileName+"... "+sizeInBytes+" uploaded..... </h1></br>");
}
```

Esta parte del código es la que se ejecuta para subir un archivo desde el navegador del cliente al servidor, a continuación se detallan los pasos seguidos.

- Se comprueba que el contenido de la petición es un archivo multipart, es decir, que el usuario está intentando subir un archivo y no pedir una página web.
- Una vez comprobado, se crea un elemento factory que gestiona los archivos que se escriben en disco en el servidor.
- A partir de este elemento factory, se crea un objeto que permite gestionar la subida de archivos a través de un servlet y se comunica con el factory, que es quien gestiona el disco en el servidor.
- Una vez creadas las clases que gestionan el disco, se procede a procesar la petición de subida de archivos.
- Los archivos a subir se tratan de uno en uno, en este caso sólo hay uno, pero la interfaz permitiría subida múltiple, primero se obtiene su nombre, para poder subirlos, después se crea un nombre basado en la fecha y hora actuales del servidor, para poder crear una copia del archivo en el servidor y almacenarla con objeto de poder revisarla posteriormente en caso de fallo.

- El archivo se escribe en el servido y se llama a la función tratarExcel que es la que interpreta los archivos.

Esta interfaz de subida de archivos es común para los tres tipos de esquemas de archivos xls con los que se trabaja, que son el esquema de añadir encargos, el de asignar horas lectivas a profesores y el de definir horarios de profesores.

A continuación se muestran algunos fragmentos de código de la función que trata estos archivos xls, para el esquema de añadir encargos de asignaturas al sistema.

```
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.hssf.usermodel.HSSFRow;
……

InputStream myxls = new FileInputStream(uploaded.getAbsolutePath());
HSSFWorkbook wb     = new HSSFWorkbook(myxls);
HSSFSheet hoja = wb.getSheetAt (0);
HSSFRow f1=null;
HSSFCell c0;
int filas=hoja.getPhysicalNumberOfRows();
```

Tal y como se ha descrito en la introducción de esta memoria, los archivos xls se manipulan mediante el uso de las librerías hssf del proyecto Apache POI. En este fragmento de código se puede observar la referencia implícita a las librerías de este proyecto, con tal de que el compilador de java pueda generar el código de la aplicación.

Tras esta inclusión de las librerías, se observa la creación del objeto que gestiona el archivo xls, que se crea pasando la ruta del archivo subido por el usuario y almacenado en el servidor. Una vez generado el contenedor del archivo, la API de programación obliga a crear un elemento workbook, que contiene todas las diferentes hojas existentes en el archivo xls original y así poder empezar a trabajar con ellas.

También se observa, en la última línea, como la API proviene herramientas para controlar, en este caso, el número de filas físicas en la hoja, siendo una fila física cualquier fila en la que se haya realizado algún cambio, que alguna de sus celdas no sean nulas.

Básicamente en estas funciones lo que se hace es ir recorriendo las filas de la hoja, mientras se obtienen sus datos. Tal y como se muestra en el siguiente código.

```
f1=hoja.getRow(cont);
String codigo= f1.getCell(0).toString(); //cogemos codigo
String siglas=f1.getCell(1).toString();//cogemos las siglas
```

Una vez recogidos los valores de la tabla, se procede a comprobarlos, si es necesario, para después introducirlos en la base de datos, como se puede observar en el ejemplo siguiente.

```
Connection con = bd.conectar();
//bucle para dar de alta los encargos
rs=bd.consulta(con, "select * from encargo e where e.siglasCentro='"+siglas+"' and
e.siglasTitulacion and e.cuatrimestre='"+cuatrimestre+"' and e.codigo='"+codigo+"'");
if(rs.next()){

bd.actualizacion(con, "update encargo e set e.gruposP='"+gruposP+"', e.gruposT='"+gruposT+"',
e.gruposL='"+gruposL+"' where e.siglasCentro='"+siglas+"' and e.siglasTitulacion and
e.cuatrimestre='"+cuatrimestre+"' and e.codigo='"+codigo+"'");
}
else{
int num2 = bd.actualizacion (con, "insert into encargo values ('"+centro+"','"+titulacion+"',
'"+cuatrimestre+"','"+codigo+"','"+gruposP+"','"+gruposT+"','"+gruposL+"' )" );
}
bd.desconectar(con);
```

Este código se ejecuta para dar de alta los encargos en la base de datos, lo primero que hace es comprobar si ya se había realizado un encargo anteriormente, con las mismas características, en ese caso lo que se hace es simplemente modificar los números de grupos de teoría, laboratorio y prácticas, existentes en la base de datos, con los indicados en el archivo xls. Si no existiera ningún encargo con las características indicadas, se procedería a crear uno nuevo.

4.2.5.- Clases de descarga de archivos

Las clases de descarga de archivos, son aquellas que permiten que el usuario se descargue archivos en formato xls, ya sea para consultar el estado actual del sistema o para poder, a partir de este archivo descargado, cargar nuevas configuraciones en la base de datos.

Cada tipo de esquema es generado por una función diferente, de las que todas ellas se agrupan dentro de una misma clase.
A continuación se muestran algunas de las principales características de estas funciones y de su implementación.

Lo que, en este caso, se envía al usuario no es una página web, si no que es un archivo xls, por lo que hay que especificar que el tipo de respuesta enviada es una archivo de aplicación xls, con tal de que el usuario pueda abrir este archivo automáticamente con el programa asignado a este formato. Esto se hace mediante la siguiente línea de código.

```
response.setContentType("application/vnd.ms-excel");
```

Con esto se consigue que el PC del usuario reconozca el archivo como una hoja de cálculo en formato xls y la pueda abrir automáticamente con el programa configurado.

El siguiente paso consiste en rellenar la hoja de cálculo con información procedente de la base de datos, para poder mostrársela al usuario.
Este proceso no es trivial, ya que en estos casos, la información a mostrar resulta de la creación de agregados, funciones matemáticas obtenidas directamente de la base de datos, o de la integración de datos de diferentes consultas. Por ejemplo en el caso de la descarga del balance de un departamento, se debe mostrar información sobre los encargos y capacidades de cada una de las secciones del departamento, lo cual no se puede obtener con una única consulta, porque si existe alguna sección sin encargo o sin capacidad, esta no aparecería, por lo que se deben hacer tres consultas, una para obtener las secciones, otra para obtener los encargos y otra para obtener las capacidades.
Este método consigue que el número de consultas enviadas a la base de datos sean mínimas, solo se necesitan tres para generar toda la tabla, en lugar de una para cada sección del departamento, aunque estas consultas sean más costosas de realizar, es más barato, en términos de tiempo, hacer una consulta pesada que ocho ligeras, ya que lo más lento es la conexión y desconexión a la base de datos.

En el siguiente fragmento de código se puede observar como se soluciona esta problemática de tener que integrar información de varias consultas, en concreto se observa el ejemplo de colocar las capacidades de cada una de las secciones en el balance del departamento.

```
Connection con = bd.conectar();/*Obtiene las capacidades de la seccion*/
 ResultSet rs = bd.consulta(con, "select sum(c.puntos), s.siglas from profesor p, contrato c,
profesorSeccion ps, seccion s where s.departamento='"+departamento+"' and ps.seccion=s.siglas and
p.DNI=ps.profesor and p.contrato=c.tipo group by s.siglas order by s.siglas");
int i=1;
while(rs.next()){
        HSSFRow filaBucle=sheet.getRow(i);
        while(!rs.getString(2).contentEquals(filaBucle.getCell(0).toString())){/*Avanza hasta encontrar
        la seccion que toca*/
                filaBucle.createCell(1).setCellValue(0);
                filaBucle=sheet.getRow(i+1);
                i++;
        }
        i++;
        filaBucle.createCell(1).setCellValue(Double.valueOf(rs.getString(1)));
}
```

La consulta suma los puntos de los contratos de todos los profesores que están en una sección que pertenece al departamento y los agrupa y ordena alfabéticamente por secciones.
El archivo xls ya se ha ido generando anteriormente, y llegado a esta paso, existe una fila para cada sección, ordenadas por las siglas de la sección, en las cuales la primera celda es el nombre de la sección, la segunda es la capacidad, la tercera el encargo y la cuarta el balance.

La consulta que se ha realizado en el ejemplo de arriba, solo obtiene secciones en las que hay profesores asignados, por lo que se deben saltar filas del archivo xls para mantener la correlación de secciones. Es por ello que dentro del bucle que recorre la consulta, se ha creado otro bucle que avanza filas del archivo xls mientras que las siglas de la sección de la consulta coincidan con las siglas de la sección de la fila analizada. Para cada sección que no aparece en el resultado de la consulta, se coloca un cero en su capacidad. En caso de coincidencia, el resultado de la consulta se escribe en la celda correspondiente.

Además de únicamente rellenar las hojas de cálculo con datos de la base de datos, también se han explotado más capacidades de las hojas de cálculo, como el uso de formulas.
Se han usado fórmulas para calcular totales, como en el caso de las asignaciones de horas lectivas a profesores, donde interesaba disponer de un total para poder colocarlo en la última columna de cada fila.
En Excel las fórmulas se definen identificando las celdas en las que actuará la fórmula, y la API utilizada proporciona un método, debajo mostrado, para añadir fórmulas a una celda.

```
row.createCell(1).setCellFormula("SUM(B2:B"+(i)+")");//crea las formulas para los totales
```

Además de las fórmulas, también se ha aprovechado la funcionalidad de formateo condicional de celdas que Excel ofrece, esto consiste en que cuando el valor de una celda cumple una condición especificada, se puedan realizar acciones sobre esa celda, como por ejemplo cambiar el color de fondo de la celda, que es lo que se ha hecho en este proyecto. El modo de implementar esa acción usando las librerías Apache POI es el siguiente.

```
HSSFConditionalFormattingRule rule =
(sheet.getSheetConditionalFormatting()).createConditionalFormattingRule(ComparisonOperator.LT,
"0", null);
 // Crear patron con fondo rojo
 HSSFPatternFormatting patternFmt = rule.createPatternFormatting();
patternFmt.setFillBackgroundColor(HSSFColor.RED.index);
// Definir una region que contenga las celdas deseadas
CellRangeAddress [] regions = { new CellRangeAddress(i,i,3,3)};
// Aplicar el formateo condicional a la region definida
(sheet.getSheetConditionalFormatting()).addConditionalFormatting(regions, rule);
```

Lo que aquí se hace es, primero definir la condición que se aplicará a la celda, en este caso se aplicará una comparación de si el contenido es menor que cero. En este caso la comparación se realiza con un cero, pero también se podría poner una referencia a una celda de la hoja de cálculo.
Tras definir la comparación, se debe definir la acción a realizar en caso de que la comparación sea afirmativa, aquí se ha optado por pintar el fondo de la celda de color rojo.

El último paso es definir el conjunto de celdas para las que se va a aplicar la condición. Para realizar esta definición, se tiene que pasar un rango de celdas, definido por la primera y la última fila del rango y por la primera y la última columna del rango. Como en este caso solo se quería aplicar el formateo a una celda, las filas y columnas origen y final coinciden. Una vez definida la región, se añade el formateo condicional especificado a la región especificada.
La interfaz de programación impone una limitación en el uso de formateos condicionales, sólo se pueden definir tres como máximo en una celda, sin embargo esta restricción no afecta al proyecto, pues como máximo se han llegado a utilizar dos formateos condicionales en una misma celda, fondo rojo si no se cumple y fondo verde si se cumple.

Por último, una vez generada la hoja de cálculo, con todos sus datos, se escribe el archivo generado en el canal de salida del servidor y se envía al usuario a través de este, tal y como se describe en las siguientes líneas de código.

```
ServletOutputStream out = response.getOutputStream();
wb.write(out);
out.flush();
out.close();
```

4.3.- CAPA DE PRESENTACIÓN

4.3.1.- Clase HTMLPrinter

En el tipo de proyecto implementado la implementación de la capa de presentación es sustancialmente diferente a la clásica interfaz de usuario que, usando Java, sería implementada mediante Swing y constaría de diferentes formularios y menús, en este caso la presentación se realiza a través de HTML, por lo que la implementación de esta interfaz de usuario es mucho más sencilla, sólo hay que preocuparse de generar un código HTML que muestre lo que tenga que mostrar y que, a ser posible, cumpla con las especificaciones del W3C, World Wide Web Consortium, que es el organismo que especifica el estándar HTML 4. Asegurando así que si un código HTML cumple las especificaciones del W3C, será reconocido por cualquier navegador que cumpla con el estándar.

En todo los fragmentos de código vistos hasta ahora, el código HTML generado simplemente se dedicaba a rellenar la parte principal de la página, la que contendría la información y los formularios necesarios. Del resto de la estructura de la página no se ha generado nada, ya que su contenido es siempre el mismo, por lo que se genera siempre al final de la ejecución, cuando se llama al controlador HTMLPrinter.

A continuación se muestran algunos fragmentos de código que ilustran la construcción de la página web final que el usuario verá cuando acceda al sistema o realice cualquier acción sobre él.

```
out.println("<body>");  //Inicio del body HTML
out.println("<div id=\"global\">");//contiene toda la pagina
out.println("<div id=\"cabecera\">");//contiene el logo y el titulo
out.println("<div id=\"logotipo\">");//podemos poner el logo de  la upc
out.println("<a href=\"http://"+ip_server+":8080/examples/servlets/servlet/principal\">");
out.println("<img src=\"../imagenes/logo.gif\" width=\"270\" height=\"120\"");
```

Este pequeño fragmento muestra la inclusión de alguna de las sentencias HTML que forman parte de la vista general de la web. En concreto se muestra la creación del cuerpo de la página y la inclusión del logotipo de la UPC, que en este caso hace de enlace a la página de inicio.

Como se puede comprobar la forma de generar el código HTML es muy similar a la realizada en las clases del dominio. La diferencia en este caso es que en lugar de escribir los fragmentos de código en un vector, se escriben directamente en el canal de salida del servidor, el canal out.

Esta clase genera la estructura de la página, los menús de enlaces, la cabecera y el pie de página.

La única variabilidad que se produce es que si se detecta que el usuario ha iniciado sesión en el servidor, se muestra su nombre de usuario y un enlace para cerrar la sesión. En caso contrario se muestra un formulario para que el usuario entre su nombre de usuario y su contraseña.

Para reflejar el contenido generado por las clases del dominio, se usa el siguiente bucle, que recorre el vector que se ha pasado en la llamada al método.

```
for(int i=0; i<contenido.size(); i++) {
        out.println((String)contenido.elementAt(i));
}
```

La manera de añadir las sentencias generadas anteriormente es la misma que se ha presentado en esta sección, pero hay que tener en cuenta que los elementos están almacenados en un vector, en forma de objetos, y antes de usarlos hay que convertirlos a String, mediante un casting de tipo.

Los formularios para introducir la información se generan con el método printpopup, de manera exacta a la de la generación de las páginas web. Por lo que no se ha incluido ningún extracto de ese código.

Para poder mostrar las ventanas pop-up, se han creado dos pequeños scripts en el lenguaje JavaScript, uno que se incluye en el código de la página principal, que permite abrir una nueva ventana, y otro script en el código de las ventanas pop-up, que permite crear un enlace que al ser clicado cierre la ventana pop-up y actualice la página desde la que se ha lanzado la ventana emergente.

A continuación se incluye el fragmento de código HTML que constituye el script que se encarga de cerrar la ventana y recargar la página que la había creado.

```
<!--
function targetopener(mylink, closeme, closeonly){
if (! (window.focus && window.opener))return true;
window.opener.focus();
if (! closeonly)window.opener.location.href=mylink.href;
if (closeme)window.close();
return false;
}
//-->
```

Este script se llama cuando se clica sobre el enlace de cerrar que aparece en la ventana, a este enlace se le ha añadido un parámetro que hace que cuando se pinche en el enlace, se llame al script. A continuación se muestra uno de estos enlaces que llaman al script.

```
<A HREF="http://localhost:8080/examples/servlets/servlet/principal?sec=3" onClick="return
targetopener(this, 1)">Volver</A>
```

El código HTML generado por el controlador de presentación, simplemente dispone los elementos en orden correcto de generación, esto es, se encarga de crear todos los elementos que aparecerán en la página y crear divisiones por secciones o estilos. Todo el estilo de la página web, es decir, los colores, los tipos de letra, la ordenación y demás parámetros, se encuentran en un archivo de hoja de estilos en cascada, llamado estilo-general.css.

Este tipo de archivos permiten definir el estilo de los elementos que se encuentran dentro de una misma etiqueta en el código HTML, identificada por su identificador o su clase. Se llaman hojas de estilo en cascada, porque la estructura de div, sigue la misma estructura del HTML, es jerárquica, dentro de una etiqueta div, puede haber varias etiquetas. Consiguiendo así que si definimos un estilo para el div principal, todos aquellos que estén dentro de él también usarán ese estilo, a no ser que lo redefinan.

Para que el uso de estos archivos sea correcto, hace falta asegurarse de que todo el código generado está correctamente encapsulado y dividido en etiquetas div de una manera adecuada y correcta.

El uso de hojas de estilo CSS permite modificar completamente el diseño visual de la página web sin tener que recompilar el servlet, simplemente modificando el archivo css, guardándolo y recargando la página, ya se verían los cambios realizados en el estilo de la página. Por lo que aumenta la facilidad de, si interesara, poder cambiar el aspecto de la página sin tener que tocar ninguna parte del código ni parar el servidor.

A continuación se muestran diferentes fragmentos de este archivo css, junto con una breve explicación.

```
#contenido a {
        font-size: 100%;
        margin-left: 20px;
        color: #3e6aac;
        display: inline;
        text-decoration: none;
        text-transform: capitalize;
        font-family: Georgia, "Times New Roman", Times, serif;
        font-size: 16px;
}
```

Para poder aplicar un estilo a un elemento, hay que especificar la etiqueta div y el tipo de elemento, en este caso el estilo se aplica sobre la etiqueta contenido a los elementos a que, en HTML, son los enlaces.

Se puede comprobar que se establecen ciertos parámetros como los márgenes, el color, la fuente del texto, el tamaño de la fuente, etc.

```
#pie p{

font-size: 100%;
color: #8b8b8b;
display: inline;
}
```

```
#pieuno{

display: block;
background-color: #486da4;

}
```

```
#linea ul{
        padding-right: 10px;
        list-style: none;
        text-align: right;
}
```

Aquí se puede observar la definición de otros elementos que servirían para marcar el estilo de los elementos de la web. El elemento p es el texto en HTML, se definen el tamaño de la fuente, el color y se establece que los distintos elementos de tipo p, se mostrarán uno al lado del otro, esto se hace con la propiedad display.

Para la etiqueta pieuno, se definen sus características generales, que, todos los elementos que se encuentren dentro de ella tendrán por defecto. Se establece un color de fondo y se ordena que los diferentes elementos de que se componga se muestren uno encima de otro, esto es por el valor block del atributo display.

Por último se definen las propiedades de una lista de elementos no ordenada, ul, dentro de la etiqueta línea, se establece que cada elemento debe dejar un hueco a su derecha de 10 pixeles, que no se añade ningún estilo de lista y que el texto se alineará a la derecha por defecto.

Con estos pequeños ejemplos se deja clara la utilidad y funcionamiento de las hojas de estilos en cascada, así como dejar claro que , sin una buena organización del código en etiquetas div, estas hojas de estilo resultan inútiles para implementar el diseño visual de una web.

5.- Configuración de los servidores

5.1.- Apache Tomcat

El servidor web se configura mediante la edición de los archivos Server.xml y web.xml. El archivo Server.xml contiene la configuración de los puertos en los que el servidor atiende y recibe peticiones. Es en este archivo donde se configuran parámetros necesarios para el funcionamiento del servidor, como los puertos en los que escucha, la dirección de los certificados digitales o el número máximo de usuarios simultáneos entre otros. Por defecto el servidor atiende peticiones http en el puerto 8080 y peticiones https en el 8443.

Una vez construido el servlet, es necesario configurar el servidor web Tomcat para que este sea capaz de ejecutar el servlet cuando reciba una petición desde internet.

Los parámetros de configuración de los servlets se almacenan en un archivo llamado web.xml, en este archivo se especifican el nombre del servlet, la clase que lo contiene y la dirección URL que habrá que utilizar para acceder a él.

En el ejemplo siguiente se puede observar la configuración del archivo web.xml para que el servidor sea capaz de ejecutar el servlet que se ha creado para este proyecto.

```
<servlet>
  <servlet-name>principal</servlet-name>
  <servlet-class>prueba.principal</servlet-class>
</servlet>
```

```
<servlet-mapping>
  <servlet-name>principal</servlet-name>
  <url-pattern>/principal</url-pattern>
</servlet-mapping>
```

5.2.- Base de datos MySQL

La única configuración necesaria para la base de datos es la especificación de un puerto, por defecto 3306, y la creación de una cuenta de usuario que permita el acceso y la modificación de las tablas creadas para este proyecto

5.3.- Servlet

Por la parte del servlet, la configuración a realizar consiste en dar, mediante un archivo de configuración, la dirección IP del servidor, el puerto en el que el servidor atiende las peticiones y la ruta del servlet así como la ruta de acceso a la base de datos, el driver JDBC de conexión a la base de datos y el nombre de usuario y contraseña para acceder a la base de datos.

Esto se ha realizado de esta manera para que, si fuera necesario cambiar el sistema gestor de bases de datos de MySQL a otro como PostgreSQL, no hubiera que recompilar el servlet, si no que con modificar el archivo de configuración indicando el paquete donde se encuentra el nuevo driver y los datos de acceso de la base de datos, fuera suficiente.

Para cargar una nueva configuración basta con modificar los archivos de configuración y reiniciar el servidor Tomcat.

6.- Resultados

6.1.- Aplicación desarrollada

El resultado del proceso de diseño e implementación descrito en esta memoria, ha dado como resultado la elaboración de una plataforma web, mediante la cual se pueden realizar los procesos de gestión de la docencia de una manera más sencilla y eficaz a como se realizaban anteriormente.

A continuación se muestran una serie de pantallas de la plataforma desarrollada. Estas capturas de pantalla simplemente pretenden mostrar como es la interacción del usuario con el sistema de una manera general, para más detalle habría que consultar el manual de la aplicación, que se encuentra en el anexo 1 de esta memoria.

6.1.1.- Pantalla de bienvenida

Pantalla de bienvenida de la aplicación

Al acceder a la plataforma web, el usuario observará la pantalla anterior, en la que aparece una barra de menús, que enlazarían a cada uno de los apartados, un formulario de login, para registrarse en el sistema y un mensaje de bienvenida. Hay que destacar que hasta que el usuario no se registre en el sistema, no podrá acceder a ninguno de los distintos apartados de la plataforma.

Cuando un usuario se registra en la plataforma se muestra esta misma pantalla, pero cambiando el mensaje de bienvenida genérico por uno personalizado con su nombre de usuario, y el formulario de registro por su nombre de usuario y un enlace para terminar la sesión iniciada en la plataforma.

La cabecera de esta página se mantiene para el resto de pantallas de la plataforma, todas muestran el mismo título y la misma barra de enlaces.

6.1.2.- Apartado de profesores

Menú de profesores

Esta captura de pantalla del apartado de profesores sirve para mostrar la estructura que siguen todos los distintos apartados de la plataforma.

Debajo de la línea gris, aparecen los filtros de información, en este caso se puede filtrar por DNI, nombre, departamento y sección. Hay dos tipos de filtros, por texto y por selección. Los filtros por texto son aquellos en los que el usuario debe introducir, por medio del teclado, el valor a filtrar, en este caso el DNI y el nombre. Los filtros por selección son aquellos que constan de un menú desplegable en el que el usuario puede seleccionar un valor simplemente clicando sobre él. Los campos dejados en blanco no se tienen en cuenta a la hora de ejecutar el filtrado, lo que implica que se puede filtrar por más de un campo a la vez.

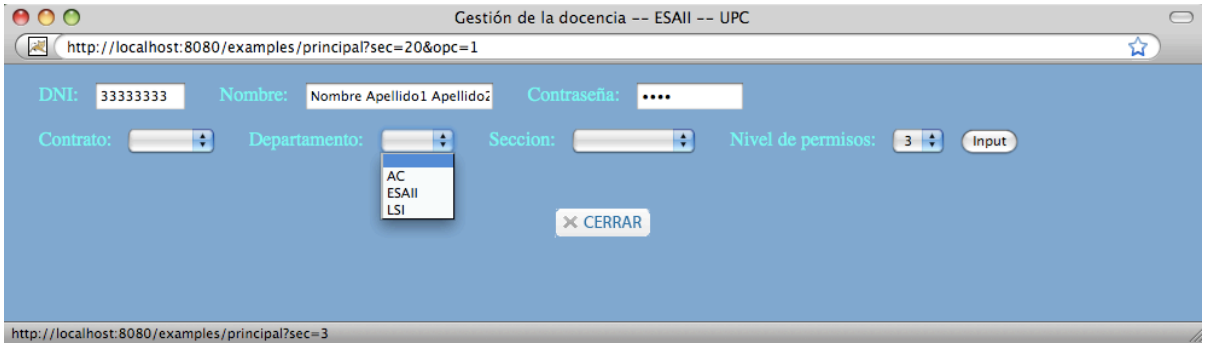
Bajo los filtros aparece la información específica de cada apartado. Esta información se muestra en forma de tabla, donde la primera línea es una cabecera que indica la información que se muestra. En este caso se muestra la información de los profesores y sus asignaciones a departamentos. La información de un profesor aparece tantas veces como asignaciones a departamentos y/o secciones tenga.

Por último, bajo la tabla de información aparecen los enlaces a las ventanas emergentes que permiten la interacción del usuario con la información existente en el sistema.

Hay 5 tipos diferentes de ventanas emergentes, de las que en este ejemplo aparecen 4, que siempre son representadas por los mismos iconos, tal y como aparecen en el apartado de profesores, modificación de contraseña, inserción de un nuevo elemento, modificación de la información de un elemento y baja de un elemento, el último tipo es la ventana de interacción con archivos xls, que se representa por un icono de Microsoft Excel.

6.1.3.- Ventanas emergentes

En este apartado se muestran dos ejemplos de ventanas emergentes que, entre las dos, contienen los diferentes tipos de formularios mediante los cuales el usuario interacciona con la plataforma.



Ventana emergente con formulario de alta de profesor

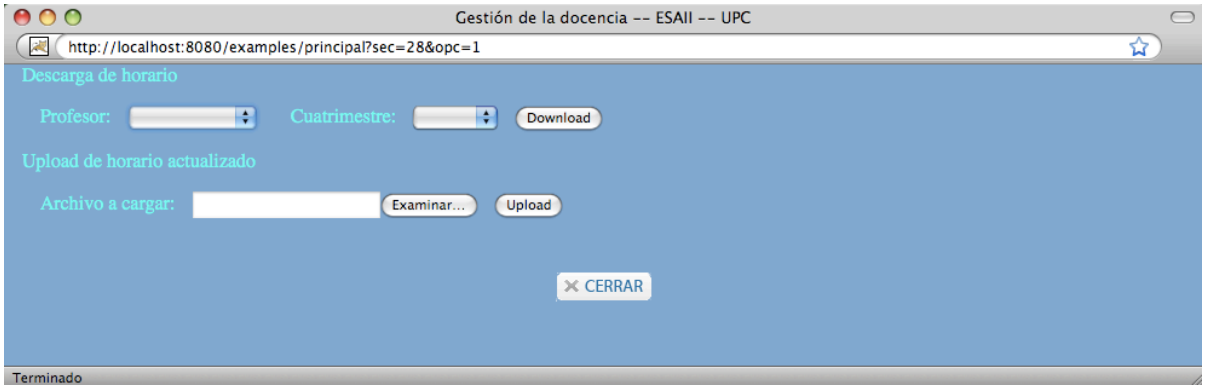
La ventana anterior es la de inserción de un nuevo profesor, pero sirve para ilustrar el tipo de formularios que se usan en el resto de ventanas.

Como se puede observar aparecen campos de texto y menús desplegables, que será la forma que el usuario usará para interaccionar con la plataforma.

Esta ventana contiene un único formulario en el cual los únicos campos obligatorios son los que aparecen en la parte superior, los demás son opcionales, ya que no son identificadores necesarios para el profesor.

Este caso muestra una ventana de inserción, los formularios de las ventanas de modificación y baja son similares a este, la diferencia es que así como en la inserción hay que definir los campos que identifican al elemento de información, DNI y nombre en este caso, en el resto de ventanas estos valores se seleccionan mediante menús desplegables.

Todos los formularios creados se han diseñado de manera que la entrada de información mediante teclado se reduzca al mínimo necesario, evitando así posibles errores ortográficos y facilitando la selección al usuario.



Ventana emergente con formularios de descarga y subida de horarios

La captura anterior muestra la ventana emergente de descarga y subida de horarios de profesores mediante archivos xls.

En este caso aparecen dos formularios diferentes, el primero sirve para descargar un archivo en formato xls, en este caso el horario de un profesor en un cuatrimestre, para lo que el usuario debe seleccionar el profesor y el cuatrimestre deseados mediante los menús desplegables. El segundo formulario sirve para cargar un archivo xls e introducir su información en la plataforma, pinchando en el botón examinar, se muestra un diálogo de selección de archivos, una vez seleccionado el archivo al clicar en el botón Upload, el archivo se carga en el sistema y se indica al usuario si la carga ha sido correcta o si por el contrario ha habido algún problema.

6.2.- Cumplimiento de los objetivos

En este apartado se detalla el grado de cumplimiento de los objetivos iniciales de proyecto alcanzados.

6.2.1.- Requisitos funcionales

El proceso de elaboración del proyecto se ha controlado el cumplimiento de todos los requisitos funcionales definidos al inicio de esta memoria, es por ello que el sistema diseñado cumple todas las funciones que se habían acordado en un principio.

Es decir, el sistema es capaz de gestionar titulaciones, centros, departamentos, secciones, asignaturas, encargos, profesores y contratos. Así como la asignación de horas lectivas a profesores, la consulta de balances e informes de departamentos y la consulta, modificación e inserción de horarios de profesores.

6.2.2.- Requisitos no funcionales

Los requisitos no funcionales son aquellos que no es necesario cumplir para que el sistema permita hacer todo lo que debería hacer, es decir, definen el cómo hacer las cosas y no el qué hacer. Es por ello que el cumplimiento de estos requisitos, al máximo grado posible, es vital para cualquier sistema de información, ya que si un sistema hace todo lo que tiene que hacer, pero no lo hace como los usuarios han pedido que lo haga, este sistema seguramente o no será usado o no será del agrado de los usuarios, lo que a la larga conllevaría a que se dejara de usar.

Es por este motivo por el que se ha hecho especial hincapié en el cumplimiento, al máximo posible, de los requisitos no funcionales.

A continuación se muestra la lista de los requisitos no funcionales definidos al principio de esta memoria con su grado de cumplimiento y una pequeña explicación de por qué se ha hecho así.

- **Coste del desarrollo y mantenimiento**

  Se buscaba que el coste del mantenimiento y desarrollo fuera el menor posible, para lo que se propuso usar una serie de tecnologías de código libre.

Este requisito se ha cumplido a la perfección, ya que todas las tecnologías propuestas se han podido utilizar y el resultado ha sido completamente satisfactorio.

Además a parte de libre, el sistema también ha resultado ser multiplataforma, ya que el servidor Tomcat está programado en Java, al igual que los servlets, por lo que el código se puede pasar de un sistema operativo a otro, sin necesidad de recompilarlo, con la única condición de disponer de una máquina virtual de Java en el sistema.

La base de datos elegida en este caso ha sido MySQL, ya que es de código abierto y tiene versiones para prácticamente cada sistema operativo. No obstante la implementación del sistema es independiente del sistema gestor de bases de datos, si se quiere utilizar otro SGBD, únicamente hay que conseguir su driver JDBC e indicar el nombre de este en el archivo de configuración del servlet, junto con la ruta de acceso a la base de datos del sistema en ese nuevo SGBD.

- **Minimización del impacto en los usuarios**

Durante la especificación de requisitos del sistema se pidió como requisito principal que permitiera el uso de archivos de Microsoft Excel para realizar encargos y asignaciones de horas lectivas y horarios de clase a profesores.
Para la realización de este requisito se decidió utilizar las librerías Apache POI, que permitían la creación y modificación de archivos xls utilizando el lenguaje Java.

El uso de estas librerías ha resultado satisfactorio, la generación y lectura de archivos xls se ha podido implementar sin problemas. Además se han podido incluir fórmulas y reglas de formateo condicional en estos archivos que, a priori, no conocía si la API de Apache POI lo permitía, que han resultado útiles para facilitar la lectura y edición de las hojas de cálculo.

Durante la utilización de estas librerías se han detectado dos problemas, descritos a continuación.

Los archivos xls generados no disponen de formato de celdas, por lo que todos se interpretan con la configuración del Excel de cada usuario, por defecto números, esto resultaba en un problema en el caso de la subida de horarios, porque al introducir las horas en el archivo descargado, se interpretaban como fechas completas y el valor leído por Java no era correcto. Este problema se solucionó creando un archivo xls vacío, cuyas celdas son de formato texto, en el servidor, cuando se ha de generar un horario, se lee este archivo, se introducen los datos y se envía al usuario, de esta manera todas las celdas se interpretan como texto y no hay problemas de conversión de formato.

El segundo problema no tiene ninguna influencia negativa en la implementación, si bien no se ha podido solucionar. Consiste en que a los archivos descargados no se les puede asignar un nombre ni una extensión, todos se llaman *principal*, por lo que será tarea del usuario asignarles un nombre. No obstante a pesar de no tener extensión el tipo MIME de los archivos está definido, por lo que aunque

no tenga extensión, son abiertos por defecto por el programa de hojas de cálculo predefinido en el sistema, ya sea Excel, OpenOffice o cualquier otro.

- **Mayor accesibilidad posible**
  Durante la especificación se acordó que el sistema debería ofrecer la máxima accesibilidad posible, por lo que se optó por crear una web utilizando la interfaz de Java servlets.

  Este requisito se ha cumplido a la perfección, ya que la interfaz de Java servlets ha ofrecido toda la funcionalidad requerida. Además la web se ha diseñado de una manera gráfica que ofreciera una interfaz lo más usable posible, para lo que se ha usado una hoja de estilos css, para definir el diseño de la web, junto con dos código javascript que permiten el lanzamiento de ventanas pop-up en las que se muestran los formularios para la interacción con el sistema.

  También se ha buscado que el usuario tuviera que introducir el mínimo de datos por teclado, por lo que todos los campos de formularios que hicieran referencia a entidades del sistema se han implementado mediante menús desplegables, lo que facilita la selección, ya que no puede haber errores de escritura.

  Un problema derivado de esta solución es que hay que confiar en que el navegador web cumpla con la especificación de HTML y CSS del W3C. Firefox, Safari y Opera la cumplen, sin embargo Internet Explorer no la cumple del todo, por lo que, algunos elementos de la página no se mostrarán perfectamente en Explorer, aunque la página seguirá siendo completamente funcional.

6.2.3.- Otros aspectos tomados en consideración

Además de los requisitos funcionales se han tomado en consideración otros aspectos, también esenciales, que no se tuvieron en cuenta a la hora de realizar la toma de requisitos.

- **Normalización del modelo de datos**
  El modelo de datos se ha diseñado de forma normal, esto es un proceso del diseño de las bases de datos que consiste en evitar la redundancia de los datos, evitar problemas de actualización y proteger la integridad de los datos.
  La adopción de este proceso implica el seguimiento de una serie de reglas durante la creación del modelo de datos que permiten respetar los principios de este modelo.
  Aunque el cumplimiento de estas reglas resulten en un mayor esfuerzo a la hora de diseñar la base de datos, facilita la creación de consultas y reduce el número de joins, productos cartesiano entre tablas, que tendrá que hacer el SGBD para realizar las consultas, lo que redunda en una mayor eficiencia de las consultas.

- **Seguridad en los accesos a la base de datos**
  La seguridad en los accesos a la base de datos básicamente consiste en evitar que se realicen ataques de inyección de SQL. Un ataque de inyección de SQL consiste en que, un usuario malicioso, introduce una sentencia SQL que pretende modificar la base de datos dentro de algún campo de texto de un formulario de la web. Esta deficiencia viene dada por el controlador de bases de datos, ya que la forma de enviar una sentencia a la base de datos, es simplemente enviar la cadena de caracteres de la sentencia, sin analizarla, por lo que el programador no tiene control de lo que el usuario pueda introducir.
  No obstante este problema tiene solución. Todos los campos de texto que introduce el usuario por los formularios, van cerrados entre comillas, de manera que la base de datos los interpreta como cadenas de caracteres y no como sentencias. Teniendo esto en cuenta, lo único que hay que hacer es analizar la cadena de caracteres que el usuario ha introducido, si esta cadena contiene alguna comilla, que sería la única forma de inyectar una sentencia SQL, se descarta.

- **Seguridad en las sesiones de los usuarios activos en el sistema**
  El primer aspecto a tener en cuenta sobre la seguridad de las sesiones de los usuarios es el login.
  El login de un usuario se comprueba con un hash de la contraseña almacenado en la base de datos, esto implica que nadie, ni siquiera el administrador de la base de datos, puede conocer la contraseña original de un usuario, como mucho podría conocer su hash, pero con él no podrá recuperar la contraseña original, lo que reduce la posibilidad de un acceso no autorizado al sistema.
  Otro aspecto a tener en cuenta es el mantenimiento de las sesiones de los usuarios en el servidor, en esta sesión se almacenan el nombre de usuario, su identificador y su nivel de accesos. Al ser gestionadas por el servidor, estas sesiones no pueden ser modificadas por el usuario, lo que podría pasar si en lugar de sesiones se usaran cookies. Por lo que un usuario no podrá realizar ninguna operación no permitida para su nivel de permisos.

- **Privacidad de las comunicaciones**
  Otro aspecto que se ha tenido en cuenta ha sido el de la protección de los datos que circulan a través de la red. Todos los datos que circulan a través de la red pueden ser escuchados por terceros y usados para, potencialmente, dañar al sistema o realizar accesos no permitidos. Es por esto que los datos más sensibles, en este caso la única información confidencial serían las contraseñas, podrían pasar por una comunicación https basada en un certificado de servidor.
  Una comunicación https consiste en que el servidor envía al cliente su certificado digital de clave pública, el cliente utiliza este certificado para encriptar la petición que le envía al servidor y el servidor la desencripta con su clave privada.
  Este sistema se basa en la encriptación RSA, mediante la cual una cadena de caracteres encriptada con la clave pública de un certificado, solo puede desencriptarse con la clave privada de ese mismo certificado.
  En la versión de prueba de este proyecto no se ha implementado el cifrado https, sin embargo si que se ha comprobado su funcionamiento.
  Para implementarlo únicamente hay que cambiar los enlaces de las vistas que transportan información sensible, como son el login y la modificación de la contraseña, únicamente habría que cambiar la dirección del enlace, http por https y el puerto web por el puerto https.

Por parte del servidor, si se quiere permitir la conexión https, hay que generar u obtener un certificado digital, utilizando la herramienta keytool de Java por ejemplo, e indicar en el archivo de configuración de Tomcat, server.xml, la ruta donde se encuentra el certificado.

7.- Planificación del proyecto

7.1- Desarrollo temporal del proyecto

La elaboración de este proyecto ha durado 78 días, del 2 de marzo de 2009 al 25 de mayo de 2009. La dedicación media diaria ha sido de 6 horas de lunes a domingo, lo que hace un total de 468 horas de trabajo invertidas en el proyecto.

La primera etapa del proyecto ha sido de análisis y toma de requisitos. Ha sido la parte más lenta, ya que el proceso de toma de requisitos es lento y se tiene que ir revisando constantemente para comprobar que cumple con lo que el usuario necesita.

La segunda parte ha consistido en el diseño del software a realizar y la creación de las pantallas de la web. El diseño consistió en la creación de una estructura de clases y de funciones que permitiera realizar aquello que el usuario había pedido de manera que cada concepto estuviera separado de los demás, consiguiendo así una estructura fácil de entender y cambiable. Las pantallas de la web se diseñaron de manera que todas compartieran el mismo estilo, para no tener que acostumbrar al usuario a diferentes interfaces, con tal de aumentar su velocidad de aprendizaje de uso de la plataforma web.

La tercera parte ha sido la dedicada a la implementación. En esta etapa se incluyen la implementación de las conexiones con la base de datos y su revisión, la implementación de la subida y descarga de archivos y su revisión. Esta etapa ha sido lenta, pues he tenido que consultar la API de Apache POI varias veces con tal de consultar como se trataban los archivos xls usando esta API.

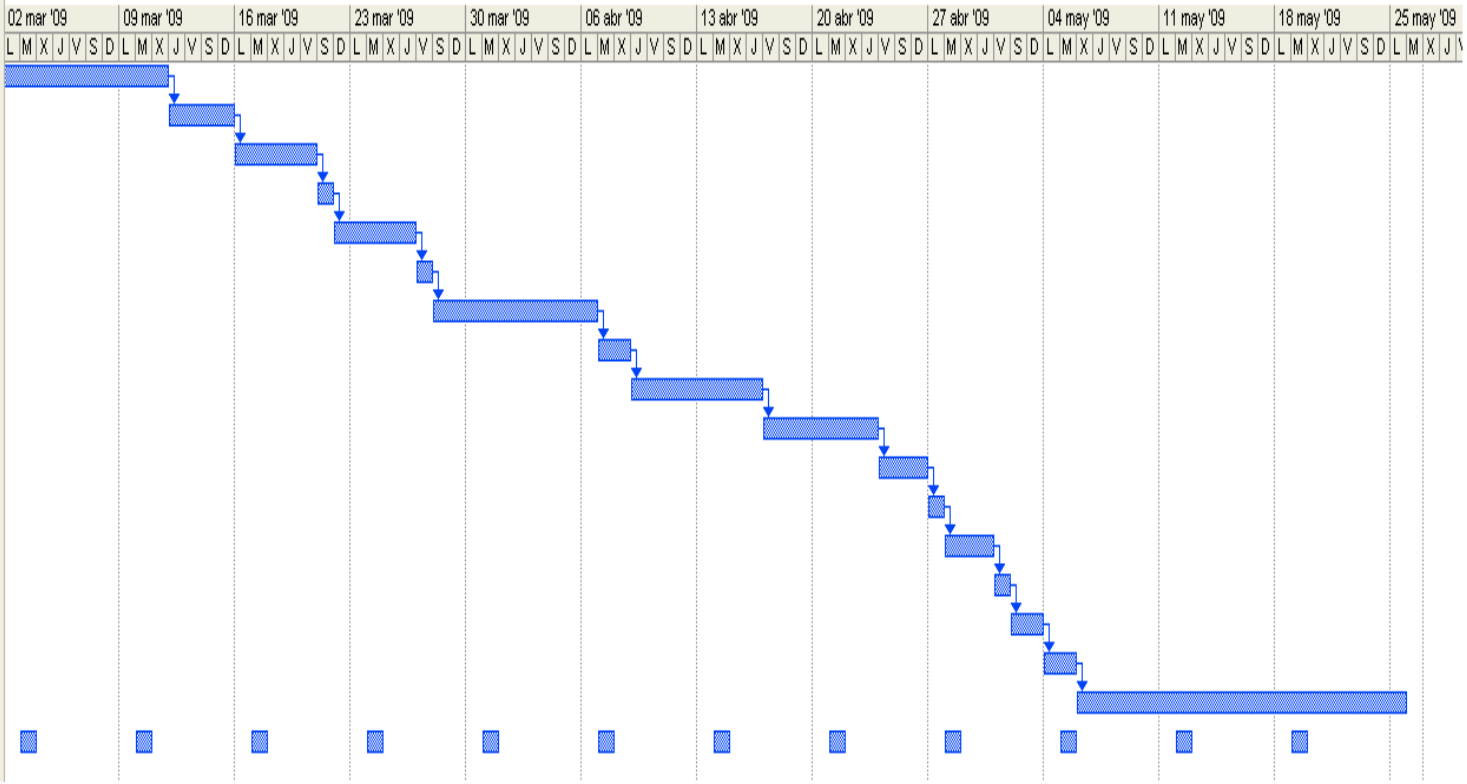
La cuarta parte ha sido dedicada a la revisión de las pantallas, y ha sido en la que se han realizado cambios, como la inserción de todos los formularios en ventanas pop-up, y la modificación de la hoja de estilos CSS para dotar a la web con un diseño más atractivo para el usuario final.

La quinta parte se ha destinado a la revisión final, en la que se descubrieron algunos errores que se tuvieron que subsanar, y a la redacción de la memoria del proyecto.

Durante toda la realización del proyecto se han realizado sesiones de control semanales con el tutor. El objetivo de estas reuniones era comprobar lo realizado, resolver dudas conceptuales y concretar el trabajo a realizar hasta la próxima semana.

En la página siguiente se adjunta un diagrama de Gantt que muestra, gráficamente, la evolución del proyecto descrita en esta sección.

Nombre de tarea	Duración
Servicio de base de datos de gestión docente	**85 días**
Diseño de la base de datos	10 días
Estructura del servlet	4 días
Página principaly hoja de estilos CSS	5 días
Revisión del servlet	1 día
Diseño del resto de pantallas	5 días
Revisión del servlet	1 día
Conexión de la base de datos con formularios	10 días
Revisión de la comunicación con la BD	2 días
Implementación de subida de archivos	8 días
Implementación de descarga de archivos	7 días
Revisión de la subida y bajada de archivos	3 días
Revisión de CSS	1 día
Formularios en pop-ups	3 días
Revisión final	1 día
Adaptación del concepto cargo	2 días
Correción del calculo de capacidad	2 días
Redacción de la memoria	20 días
Seguimiento con el director	**78 días**



8.- Presupuesto del proyecto

A la hora de elaborar un presupuesto de un proyecto informático, hay que tener en cuenta tanto las horas de trabajo del equipo del proyecto, el coste del software necesarios para que el programa creado funcione y el coste del mantenimiento requerido por el sistema.

El formato xls, usado nativamente por Excel, es un formato libre, lo que hace que la mayoría de los programas de hojas de cálculo sean capaces de leerlo y escribirlo, por lo que el uso de este tipo de formato no obliga a usar Microsoft Excel, si no que se pueden usar otras hojas de cálculo, como Open Office, que son de código abierto y están disponibles para gran variedad de sistemas operativos.

Este proyecto se ha realizado pensando en la utilización de software libre, por lo tanto puede ser ejecutado tanto sobre Linux como Windows, por lo que se presentan dos alternativas de presupuesto, teniendo en cuenta estas dos posibilidades.

Las horas de programador se han definido basándose en la planificación de trabajo que se encuentra en el diagrama de Gantt del apartado anterior.

Las horas de mantenimiento presupuestadas únicamente contemplan la gestión de copias de seguridad de la base de datos y la comprobación del correcto funcionamiento del servidor y de la lectura del registro del mismo en caso de fallo.

Concepto	Coste unitario	Unidades	Total
Hora de programador	40.00€	468	18720.00€
Apache Tomcat	0.00€	1	0.00€
SGBD MySQL	0.00€	1	0.00€
SO Ubuntu 9.04 Server ed.	0.00€	1	0.00€
SO Windows 2008 Server Web ed.	329.00€	1	329.00€
Microsoft Excel 2007 Home ed.	105.00€	1	105.00€
Open Office 3.1.0	0.00€	1	0.00€
Hora de mantenimiento	20.00€	1 por semana	20.00€

Total con Ubuntu 9.04 Server Edition y Open Office → **18720.00**

Total con Windows Server 2008 Web Edition y Open Office → **19049.00€**

Total con Windows Server 2008 Web Edition y Excel 2007 Home ed.→ **19154.00€**

Mantenimiento → **20.00€ por semana**

9.- Conclusiones

Tras haber finalizado el proyecto he comprendido todo el esfuerzo que conlleva el proceso de toma de requisitos, especificación, diseño, implementación y pruebas del proyecto. Hasta ahora los proyectos que había realizado eran en asignaturas de la facultad, en las que el tamaño del proyecto era más o menos grande, pero los entornos eran más controlados y no surgían problemas o situaciones reales.

Una de las partes en las que más he aprendido ha sido durante la especificación del sistema, es donde he comprendido que es el diseñador del sistema quien debe comprender lo que el usuario pide, incluso lo que no pide explícitamente, y ofrecerle una solución que contemple todo aquello que necesita.

También me he dado cuenta de la importancia de trabajar cerca del usuario, pues es el usuario quien va a utilizar la aplicación, por lo tanto deberá estar diseñada a su gusto, y a veces un botón colocado en un lugar, donde a priori no se buscaría, porque no se ha consultado al usuario, puede conllevar que una funcionalidad deje de utilizarse y que, en última instancia, la aplicación no sea del agrado del usuario, simplemente por una interfaz mal diseñada.

El diseño del modelo de datos también ha tenido su complicación, principalmente porque hubo algunos conceptos que durante la toma de requisitos no se tuvieron en cuenta y se descubrieron en este punto, donde se pudieron analizar e incluir en el modelo de datos, por lo que no tuvieron demasiado impacto.

El diseño de la base de datos fue bastante rápido, ya que durante la carrera he cursado varias optativas de bases de datos y es una de las áreas que más me interesan, por lo que ya conocía las reglas que había que aplicar y las situaciones a evitar, con tal de realizar un buen diseño.

Con el sistema gestor de bases de datos tuve algún pequeño problema al principio, ya que estaba más acostumbrado a trabajar con Oracle y no conocía la sintaxis exacta de MySQL, pero con consultar el manual de MySQL fue bastante. Los problema que si que tuve con MySQL vinieron a raíz del uso de la interfaz gráfica de MySQL. Existen dos aplicaciones oficiales, que son interfaces gráficas de MySQL, una para realizar consultas y otra para administrar el servidor. Resulta que estas aplicaciones no funcionan completamente y fallan al realizar creaciones de tablas, por lo que hubo que hacer todas las creaciones a mano, pero el principal problema fue el tiempo que se tardó en descubrir este fallo del programa.

El diseño del software en este caso era diferente a la programación de aplicaciones, la capa de presentación la gestiona el navegador, por lo que en ese aspecto la única preocupación era la generación de un código HTML correcto, el principal énfasis estaba en la capa del dominio, donde había que diseñar consultas a la base de datos que mostraran exactamente la información que el usuario necesitaba.

Antes de comenzar este proyecto ya había trabajado con Java, por lo que conocía los aspectos principales de este lenguaje. Ha sido con el uso de la interfaz de servlets con lo que más he aprendido, pues la he usado para todo el proyecto y he aprendido a cargar configuraciones, crear sesiones y enviar y recibir archivos a través de un servlet, lo que ha resultado de horas de lectura de manuales, pero ha sido gratificante ver los resultados.

Este proyecto ha sido la primera vez que he incluido controles de seguridad en el código, ya sea con el login, la comprobación de la inyección de SQL o la gestión de niveles de permisos. Me ha hecho descubrir que la seguridad es parte importante de cualquier proyecto de software, que nunca debe descuidarse, a no ser que se quiera dejar un sistema al libre albedrío de los usuarios y de los potenciales atacantes, lo que podría acabar en una situación completamente caótica.

La etapa de pruebas se iba realizando a la vez que la implementación, lo que resultaba en probar una funcionalidad cada vez que se implementaba, lo que ha resultado en que cuando se creaba una funcionalidad que dependía de otras, se han tenido que modificar funcionalidades ya implementadas, lo que implicaba repetir este proceso. Sin embargo creo que, aunque quizás haya sido un poco más largo que revisarlo todo al final, de esta manera se ha tenido más control sobre la implementación y siempre se ha tenido un prototipo para enseñar al usuario, lo cual ha servido para ir comprobando el grado de aceptación del sistema.

En general ha sido una experiencia enriquecedora de la que he aprendido muchas cosas y he podido poner en práctica muchos conceptos y conocimientos que he obtenido durante la carrera, lo que ha resultado de gran motivación para mí.

Anexo 1: Manual de la aplicación

1.- Acceso al sistema

Para acceder al sistema, únicamente se necesita un navegador web ya sea Mozilla Firefox, Internet Explorer, Safari o cualquier otro que cumpla los estándares del W3C. Para el correcto funcionamiento de la página web, el javascript no debe estar bloqueado por el navegador, ya que de esa manera, los menús pop-up no se mostrarían.

2.- Página principal

Una vez se accede a la web, se muestra la página de inicio, en la que aparecen los enlaces principales, un mensaje de bienvenida y el formulario de login, tal y como se muestra en a siguiente captura de pantalla.

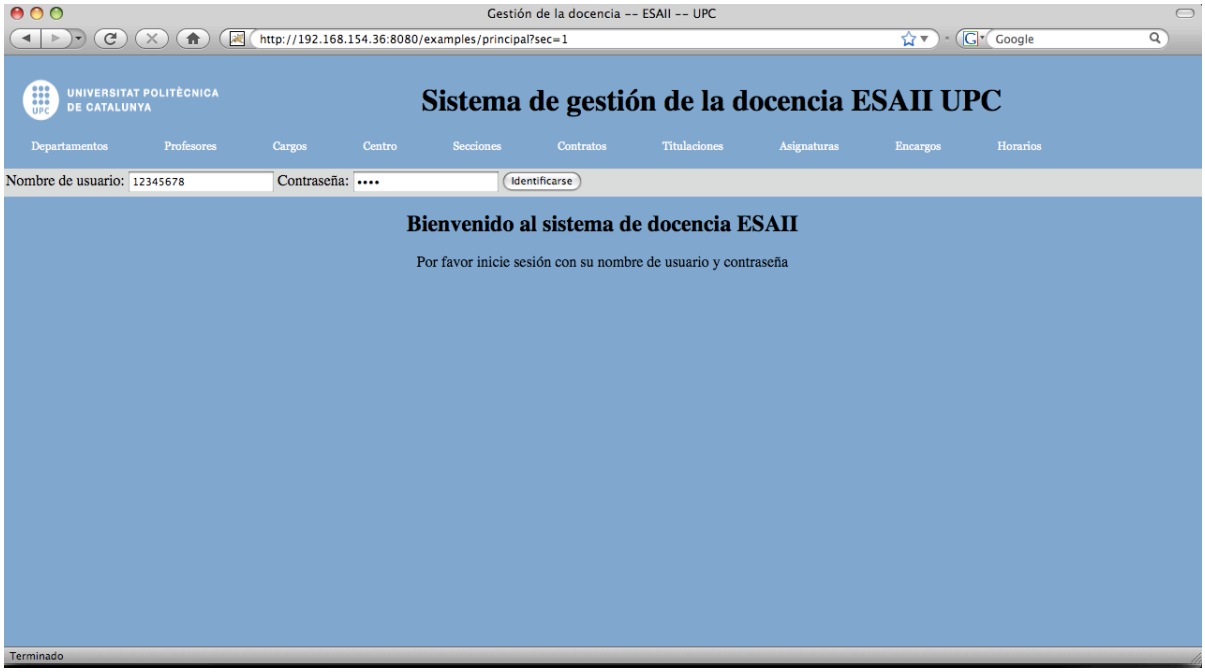


Figura 1 -- Página principal de la web

2.1.- Login

Tal y como se puede observar en la figura 1, debajo de la barra de enlaces, aparece una franja gris que contiene el formulario de login. Es en ese formulario donde el usuario debe introducir su nombre de usuario y contraseña si desea acceder al sistema.
El nombre de usuario es el DNI de cada usuario, la contraseña se define en el momento de crear un nuevo profesor y puede ser cambiada por el usuario en cualquier momento.

Figura 2 Ampliación del formulario de login

Tras clicar en el botón identificarse del formulario de login, en la figura 2, si el login es incorrecto aparece un mensaje que lo indica en la misma página principal. Si por el contrario el login es correcto, se vuelve a mostrar la página principal, pero en lugar del formulario de login aparece el nombre del usuario y un botón que permite realizar el proceso de logout, además de un mensaje de bienvenida, como se muestra en la figura 3.

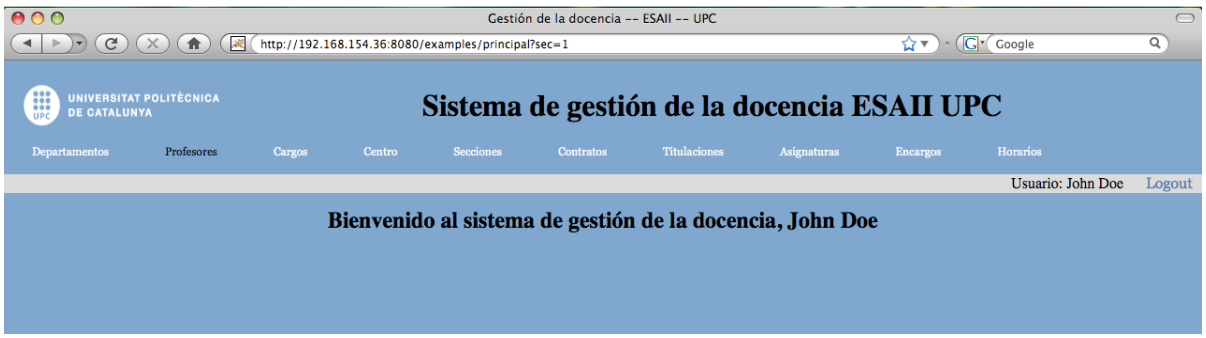


Figura 3 – Login correcto

Figura 4 – Login incorrecto

3.– Vista de departamentos

En la página departamentos se proporciona información sobre los diferentes departamentos que existen actualmente en la base de datos, así como se añaden enlaces a menús pop-up que permiten añadir, modificar y/o eliminar departamentos así como la descarga de archivos en formato Excel. Como se puede observar en la figura 5.

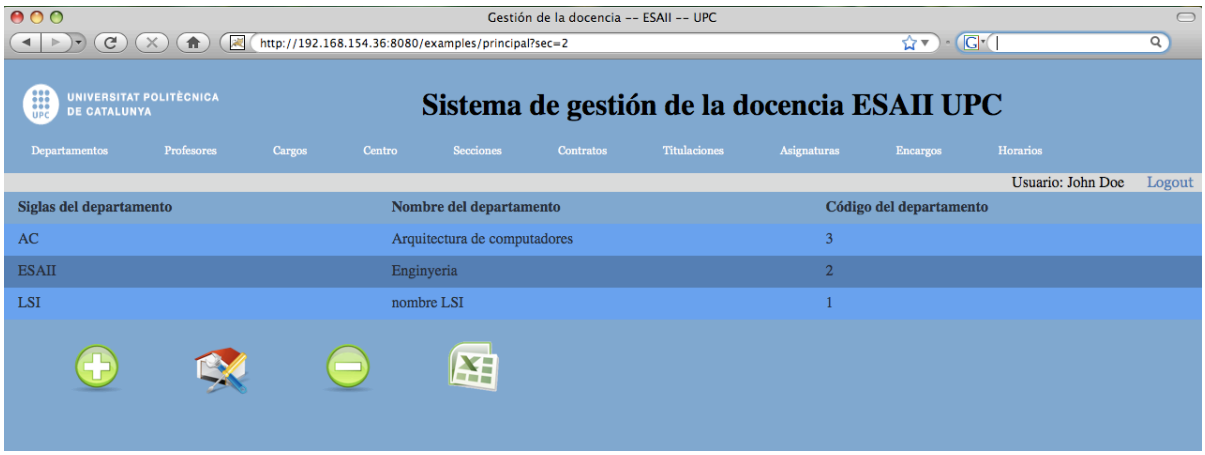


Figura 5 – Vista de departamentos

Como se puede observar en la figura 5, la vista de departamentos ofrece información sobre las siglas de los departamentos, su nombre y el código de dicho departamento.

A continuación se describen las ventanas emergentes, o pop-ups, que aparecen cuando se clica sobre alguno de los iconos que aparecen debajo de la información de los departamentos.

Estos iconos sirven para añadir nuevos departamentos, modificar departamentos existentes, eliminarlos y para descargar archivos en formato Excel que mostrarán información extra sobre los departamentos.

3.1.- Alta de departamento

Al clicar sobre el botón correspondiente, se muestra la ventana que aparece en la figura 6.
Esta ventana contiene un formulario mediante el cual se introducen los parámetros necesarios para crear un nuevo departamento. Cuando se clica en el botón introducir, se inserta el departamento en la base de datos y se muestra el mismo formulario.

Si se cierra la ventana usando el enlace cerrar ventana, se muestra la página de departamentos con la información actualizada, si no hay que actualizar la página manualmente.

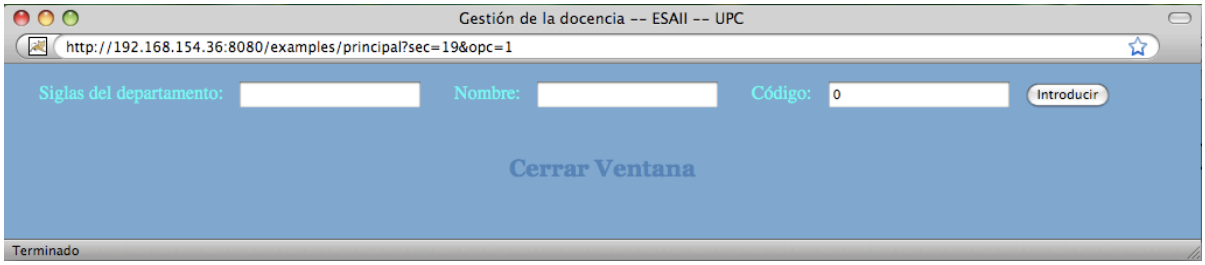


Figura 6 Pop-up de alta de departamento

3.2.- Modificación de departamentos

Esta ventana, mostrada en la figura 7, contiene un formulario mediante el cual se pueden modificar los diferentes departamentos existentes en el sistema.
El departamento a modificar se elige mediante el menú desplegable que aparece en la figura 7 y los datos a modificar se introducen en el formulario. Los campos dejados en blanco no son modificados por el sistema.

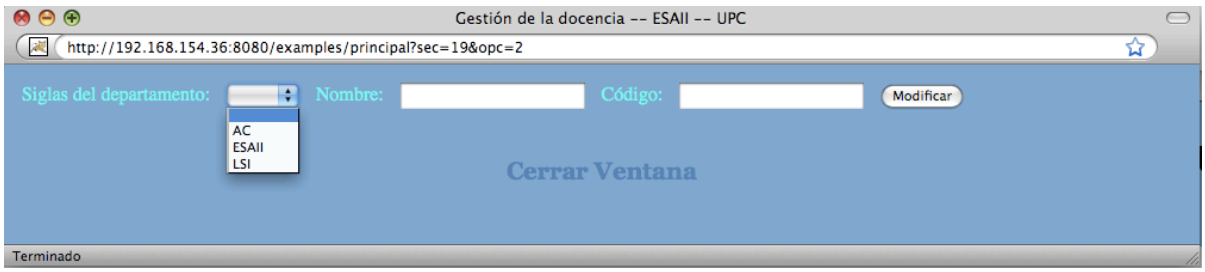


Figura 7 Pop-up de modificación de departamentos

3.3.- Baja de departamentos

La ventana que se muestra contiene un formulario en el que hay un menú desplegable, idéntico al de la figura 7. Para eliminar un departamento hay que seleccionarlo en el menú y pulsar el botón eliminar.

3.4.- Descarga de archivos en formato xls

Desde esta ventana, mostrada en la figura 8, se ofrece al usuario la posibilidad de descargarse el informe o el balance del departamento, ambos en formato xls.
Aparecen dos formularios, en ambos se piden los mismos campos. Se deben elegir el departamento y el cuatrimestre deseados en menús desplegables como los mostrados anteriormente.

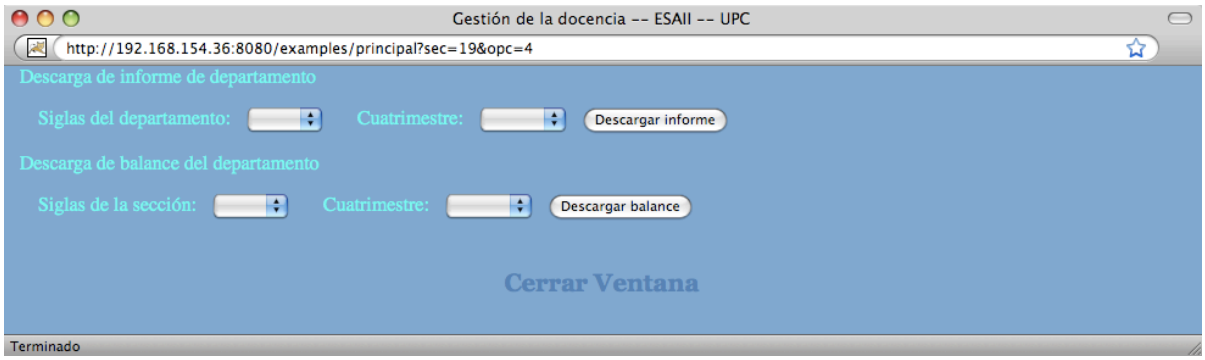


Figura 8 Pop-up de descarga de archivos xls

4.- Página de profesores

La vista de profesores muestra información sobre los profesores, en concreto se muestra su DNI, su nombre, el tipo de contrato que tienen, y sus diferentes asignaciones a departamentos y secciones dentro de esos departamentos.

Figura 9 Vista de profesores

En la parte superior de la figura 9, bajo el menú de enlaces, se observa un formulario, cuyo objetivo es el de filtrar lo profesores de los que se quiere obtener información. Este filtro dispone de varios parámetros para filtrar, el filtro aplicado finalmente estará compuesto por todos aquellos campos que no se dejen en blanco, pudiendo filtrar por más de un parámetro a la vez.

Para cada profesor se muestra una línea para cada sección a la que pertenece, por lo que si un profesor pertenece a varias secciones, su información aparece tantas veces

como asignaciones tiene. Si un profesor no pertenece a ningún departamento ni sección, solo aparecen su nombre y DNI.

En la parte inferior de la vista aparecen cuatro iconos que, al ser clicados, muestran diferentes ventanas con formularios, para, por orden, modificar la contraseña del usuario, insertar nuevos profesores, modificar datos de profesores existentes y eliminar profesores.

4.1.- Modificación de la contraseña

El primero de los iconos que aparecen en la parte inferior de la vista de profesores, véase la figura 9, lanza la ventana que permite que un usuario modifique su contraseña. En la ventana que aparece, que se puede observar en la figura 10, aparece un formulario con dos campos de texto. El usuario que desee modificar su actual contraseña debe introducir la nueva contraseña en los dos campos de texto y clicar en el botón modificar contraseña. Una vez clicado el botón aparecerá un mensaje que dirá si la contraseña se ha modificado correctamente o si por el contrario ha habido algún problema durante el proceso.

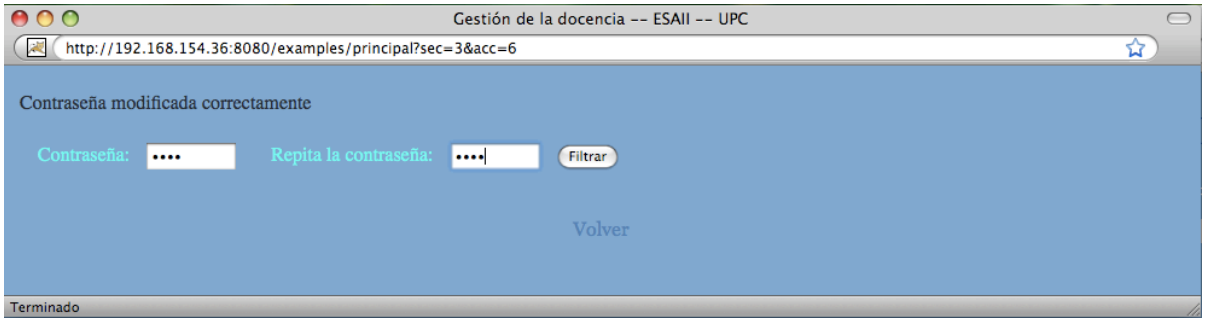


Figura 10 Pop-up de modificación de contraseña

4.2.- Alta de profesor

El pop-up de alta de profesor, figura 11, muestra un formulario que permite la introducción de un nuevo profesor en el sistema.

Tal y como se puede observaren la figura 11, algunos de los campos, como el DNI, deben introducirse manualmente, mientras que otros, como el tipo de contrato o el departamento, se eligen de una lista desplegable.

Cabe destacar que no es necesario rellenar los campos de departamento y sección, se pueden rellenar más tarde, en la ventana de modificación, si se desea.

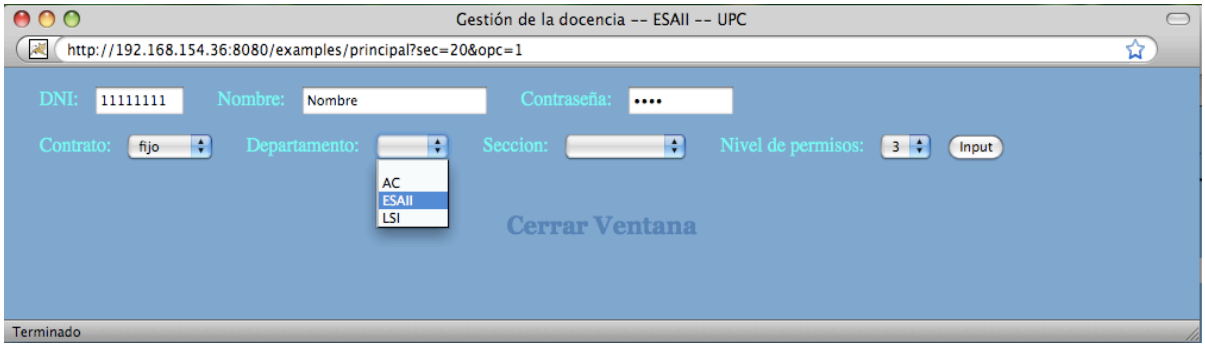


Figura 11 Pop-up de alta de profesor

4.3.- Modificación de profesor

El pop-up de modificación de datos de profesores contiene dos diferentes formularios, cada uno para modificar diferentes datos de profesores.

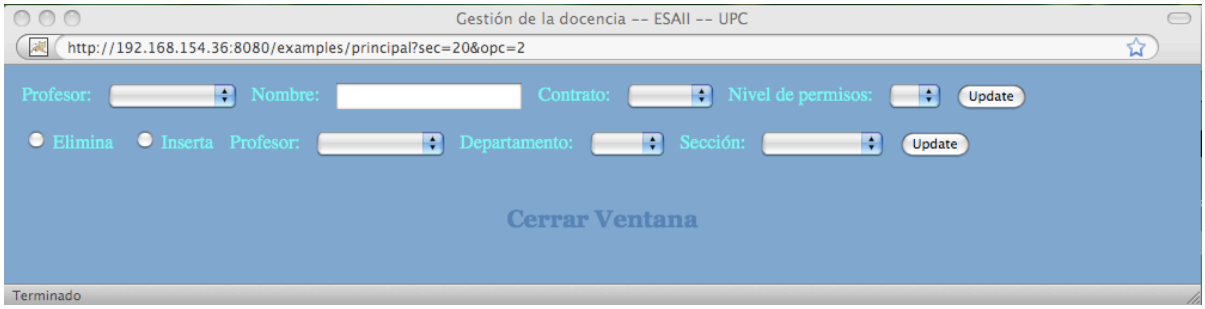


Figura 12 Pop-up de modificación de datos de profesores

El primer formulario, permite modificar el nombre del profesor, su tipo de contrato y su nivel de permisos de acceso, tal y como se puede observar en la figura 12. Los campos que se hayan dejado en blanco no se modifican al pulsar el botón del formulario.

El segundo formulario permite añadir o eliminar asignaciones de profesores a departamentos o secciones. Que se ejecute una inserción o una eliminación dependerá de la opción que se seleccione, ya sea la opción inserta o bien la opción elimina. Mientras un profesor pertenezca a más de una sección de un mismo departamento no se le podrá eliminar de ese departamento.

De la misma manera que en el resto de los formularios, los campos dejados en blanco no se tienen en cuenta.

4.4.- Baja de profesor

El formulario para dar de baja a un profesor, simplemente consta de una lista desplegable mediante la cual se selecciona el profesor a eliminar del sistema.

5.– Vista de cargos

La vista de cargos ofrece información sobre los cargos que desempeñan los profesores. En concreto se muestran el nombre del cargo, el profesor que lo desempeña, la descarga de puntos que ese cargo supone, la fecha de alta del cargo y la fecha de baja, si es que existe.

Como se puede observar en la figura 13, existe un filtro que permite que se muestren los cargos que posee un único profesor.

De la misma manera que las demás vistas, los enlaces a los formularios de alta, baja y modificación se encuentran en la parte inferior de la ventana y están representados con los mismos iconos que en las vistas ya detalladas anteriormente.

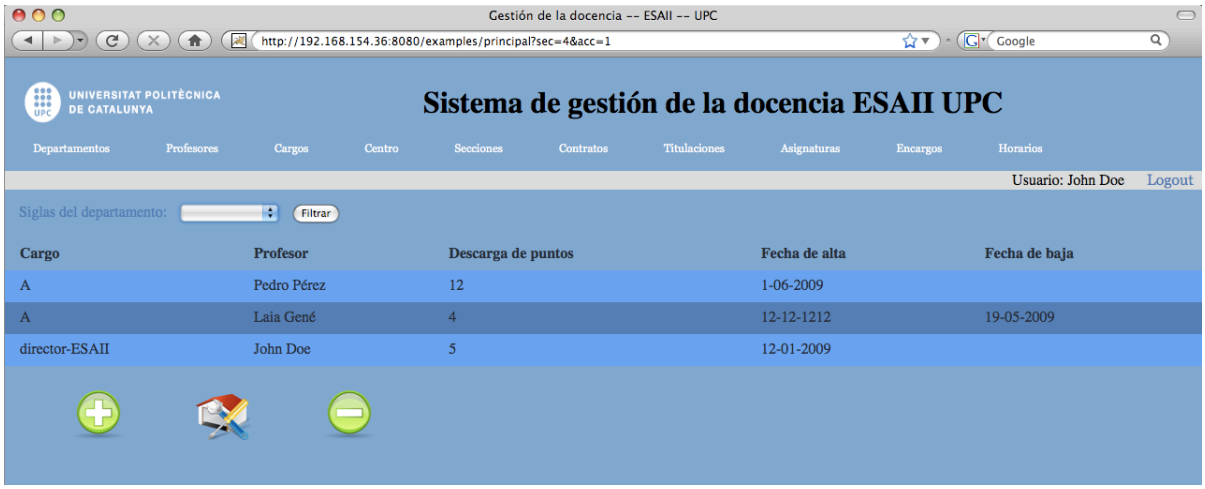


Figura 13 Vista de cargos

5.1.- Alta, baja y modificación de cargos

Los formularios de alta, baja y modificación de cargos son muy similares a los ya descritos para las vistas detalladas anteriormente, por lo que solo se detallará, en este manual del usuario el formulario de alta, disponible en la figura 14.

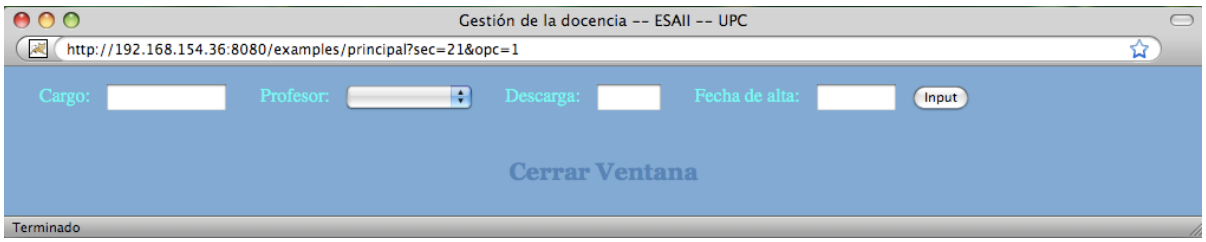


Figura 14 Pop-up de alta de cargo

En este formulario de alta, se debe definir el nombre del cargo, seleccionar al profesor que lo ocupará mediante una lista desplegable, la descarga de puntos que supondrá el cargo para ese profesor y la fecha de alta del cargo, en formato dd-mm-aaaa.

En el formulario de modificación de un cargo, se puede modificar la descarga y dar una fecha de baja, siguiendo el mismo formato que el de la fecha de alta, dd-mm-aaaa.

6.– Vista de secciones

La vista de las secciones ofrece información sobre las secciones, da las siglas de las secciones, los departamentos a los que pertenecen, el centro al que están adscritas, su capacidad lectiva actual y, si el usuario ha introducido un valor en el filtro de cuatrimestre, en la parte superior de la figura 15, también muestra el encargo para esa sección en el cuatrimestre seleccionado.

La capacidad y el encargo de las secciones se calculan a partir de la información de los profesores asignados al departamento y los encargos de asignaturas almacenados en el sistema, por lo que esta información no es modificable desde esta vista.

Si una sección no tiene profesores asignados, únicamente aparecen sus siglas, su departamento y su centro.

En la parte inferior de la vista, se muestran los iconos que enlazan a los formularios de alta, baja y modificación de secciones y , en este caso, aparece un cuarto icono que abre el menú de interacción con archivos xls.

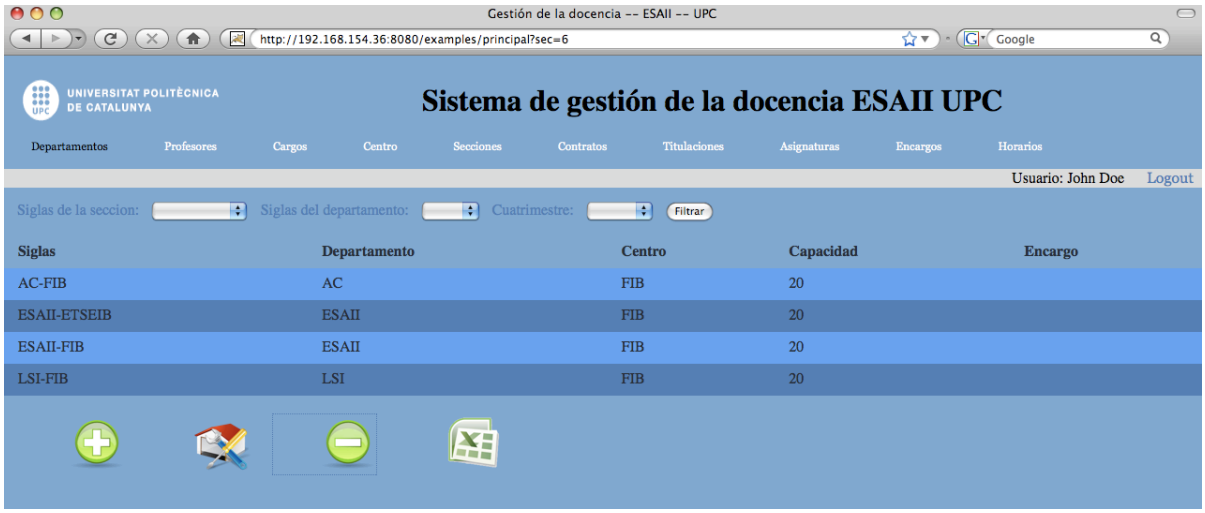


Figura 15 Vista de secciones

6.1.- Alta, baja y modificación de secciones

Los formularios de alta, baja y modificación de secciones siguen los mismos esquemas que los definidos para otras vistas.
El formulario de alta pide las siglas de la sección, el departamento al que pertenece y el centro al que está adscrita.

El formulario de modificación pide las siglas de la sección, mediante una lista desplegable, y permite modificar el centro y el departamento de esa sección, también mediante listas desplegables.

Por último el formulario de eliminación, únicamente necesita que el usuario indique las siglas de la sección a eliminar.

6.2.- Interacción con archivos de Microsoft Excel

A nivel de sección se pueden definir los puntos de horas lectivas que cada profesor de la sección debe impartir para cada una de las asignaturas que esa sección imparta en un cuatrimestre.
Para realizar esta función se han definido tres interacciones diferentes mediante el uso de archivos de Microsoft Excel. Cada una de estas interacciones se realiza mediante los formularios que aparecen al clicar el icono correspondiente en la vista de secciones. Los formularios se pueden observar en la figura 16.

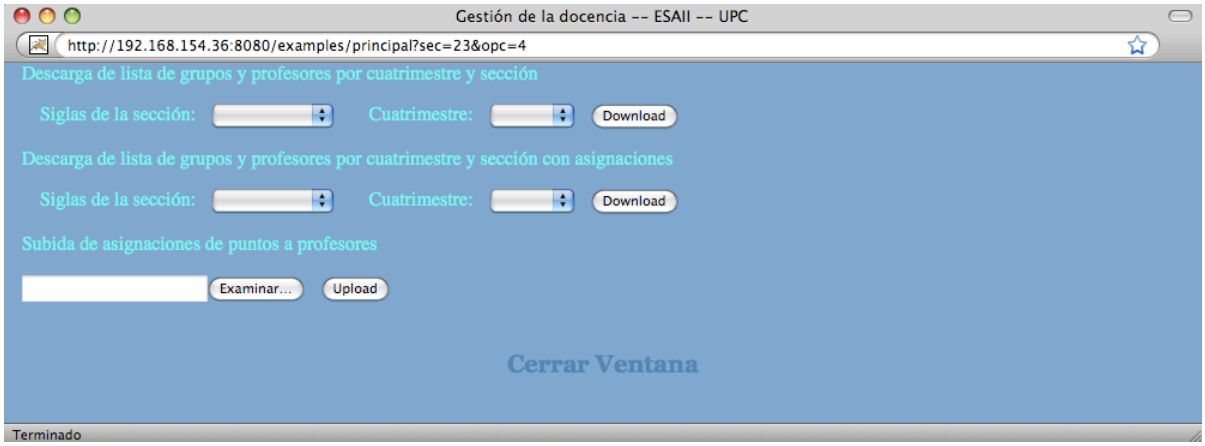


Figura 16 Formularios de interacción con archivos xls

El primer formulario sirve para, dadas una sección y un cuatrimestre, descargar un archivo xls que lee el encargo de asignaturas de esa sección en ese cuatrimestre y que contiene una lista de grupos, en función del número de grupos del encargo y la lista de profesores asignados a la sección. Usando este archivo xls, que sigue el esquema del de la figura 17, el usuario puede asignar horas lectivas a diferentes profesores en diferentes grupos.

Hay que remarcar que para la generación de este primer formulario solamente se ha leído el encargo de la base de datos. Los nombres de los grupos se han generado automáticamente, el usuario puede cambiarlos, si lo desea, de manera que estos nombres se graben en el sistema y puedan ser consultados posteriormente.

0708T	ESAII-FIB	EI	FIB			
Asignatura	Grupo	Puntos	John Doe	Ana Martínez	Pau Ferré	Total asignado
NV	T1	4.5				0
NV	T2	4.5	4.5			4.5
NV	T3	4.5				0
NV	P1	9		10		10
NV	P2	9				0
NV	P3	9				0

Figura 17 Archivo descargado con lista de grupos y profesores

Como se puede observar en la figura 17, en la primera fila del archivo aparecen unos datos que son generados por el sistema y servirán para la posterior carga del archivo, por lo que el usuario no los debe modificar.

La siguiente fila que aparece es simplemente una cabecera, que contiene los nombres de todos los profesores de la sección.

Tras estas dos filas, se genera una para cada grupo de cada asignatura que haya en el encargo, de manera que se puedan asignar horas a los profesores, indicando el número de horas en la columna que pertenece al profesor. En la última columna aparece un total de horas asignadas por grupo, si este es menor o igual que el número de horas de ese grupo, la celda aparece en verde, si es mayor la celda aparece en rojo.

El siguiente formulario sirve para descargar un archivo igual que el de la figura 17, pero con la diferencia de que en este archivo se leerían las asignaciones de horas a profesores que se hubieran guardado en el sistema anteriormente.

El último formulario sirve para cargar uno de los archivos Excel descargados en uno de los dos formularios anteriores y grabar las asignaciones introducidas en estos archivos en el sistema. Al clicar en el botón examinar se muestra un diálogo que permite buscar el archivo deseado y al clicar el botón upload, el archivo se carga en el sistema.

7.- Vista de contratos

La vista de contratos ofrece información sobre los diferentes tipos de contrato que existen en el sistema.
Un contrato se identifica por su tipo y de él se indican el número de horas y puntos que conlleva, así como su salario básico. Tal y como se puede observar en la figura 18.

La inforrmación a consultar se puede filtrar por el tipo de contrato, si solo se desea ver la información de uno de ellos.

En la parte inferior de la figura 18 aparecen los iconos que, al ser clicados, lanzan las ventanas de alta, baja y modificación de contratos.

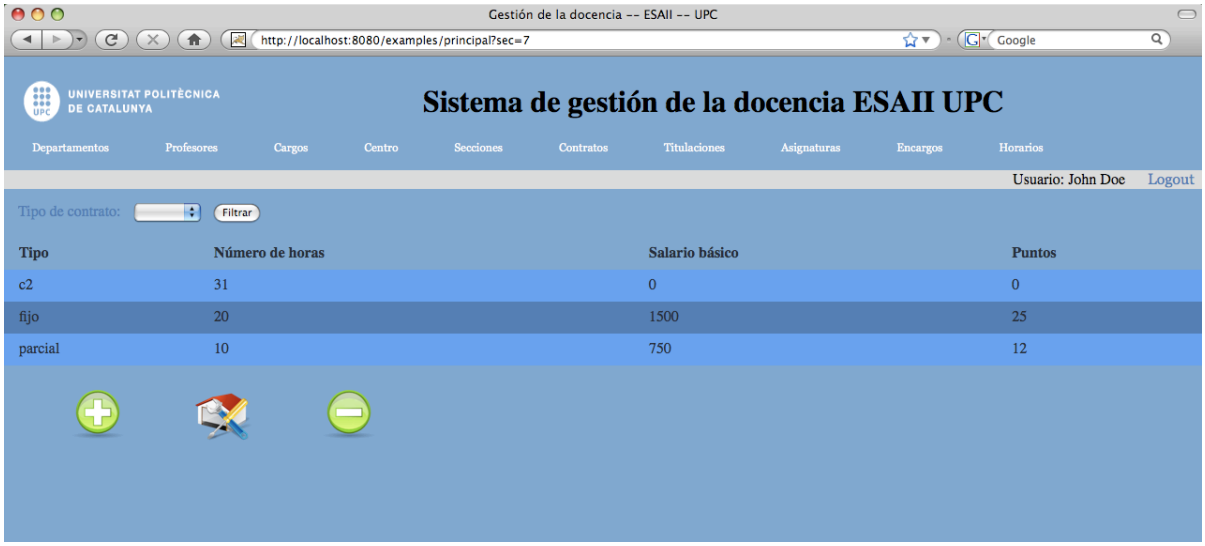


Figura 18 Vista de contratos

7.1.- Alta, baja y modificación de contratos

Las operaciones de alta, baja y modificación de los diferentes contratos se realizan de manera idéntica a la explicada en las demás vistas.
La ventana de alta pide que se rellenen los diferentes datos, mediante campos de texto.
La ventana de modificación necesita que se seleccione el tipo de contrato, para indicar que contrato se modifica, mientras que los campos a modificar se introducen mediante campos de texto, como se puede observar en la figura 19.

La ventana de baja de contrato necesita que se seleccione un tipo de contrato para darlo de baja. Hay que tener en cuenta que si existe algún profesor que tiene vigente un contrato, ese contrato no se podrá eliminar, ya que las capacidades y balances de secciones y departamentos, se calculan usando información de los contratos.

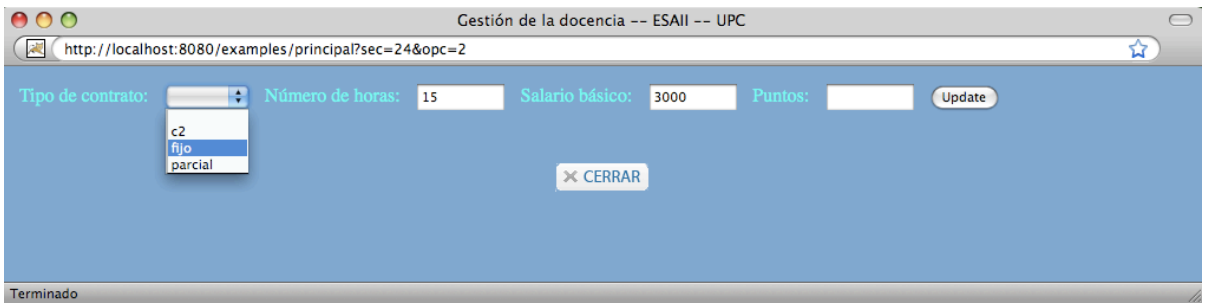


Figura 19 Ventana de modificación de contrato

8.- Vista de titulaciones

La vista de titulaciones muestra información sobre todas las titulaciones almacenadas en el sistema, en concreto muestra las siglas, el nombre, los créditos y los centros en los que se imparte esa titulación.

De la misma manera que en el caso de la vista de profesores, como una titulación puede impartirse en varios centros, en este caso aparece una fila por cada centro que imparte la titulación. Si una titulación no se imparte en ningún centro, la columna centro aparece vacía, como se puede apreciar en la figura 20.

Figura 20 Vista de titulaciones

8.1.- Alta, baja y modificación de titulaciones

Los procesos de alta, baja y modificación de titulaciones se realizan de manera idéntica a la descrita en el resto de apartados.

La única peculiaridad es en el caso de la modificación, donde aparece la opción de eliminar o añadir centros a la titulación. Este proceso se realiza seleccionando una de las dos opciones, elimina o inserta, y seleccionando la titulación y el centro mediante menús desplegables.

9.- Vista de asignaturas

La vista de asignaturas ofrece información sobre las asignaturas impartidas por la sección seleccionada mediante el menú desplegable que aparece en la vista, como se puede observar en la figura 21.

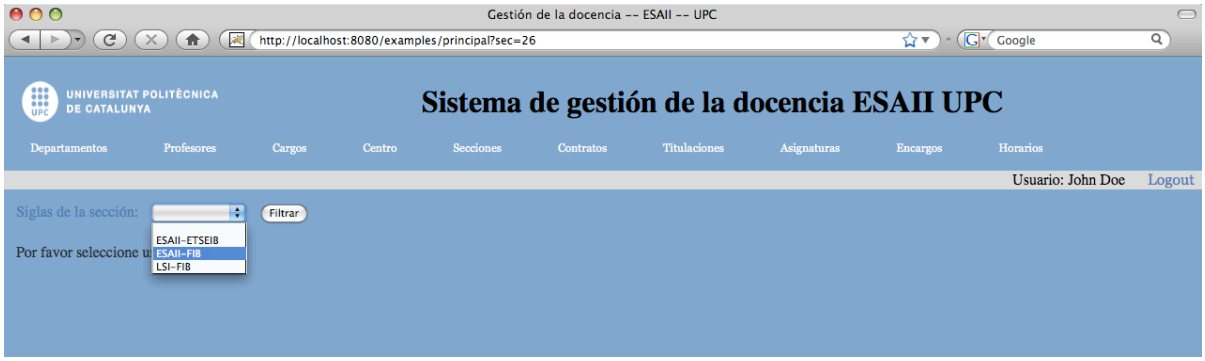


Figura 21 Vista de asignaturas sin seleccionar ninguna sección

Si no se ha seleccionado ninguna sección, se muestra un mensaje que invita al usuario a seleccionar una en concreto.

Una vez que el usuario ha seleccionado una sección, se muestra la información de las asignaturas impartidas por esa sección, como se puede observar en la figura 22, donde se ha seleccionado la sección ESAII-FIB.

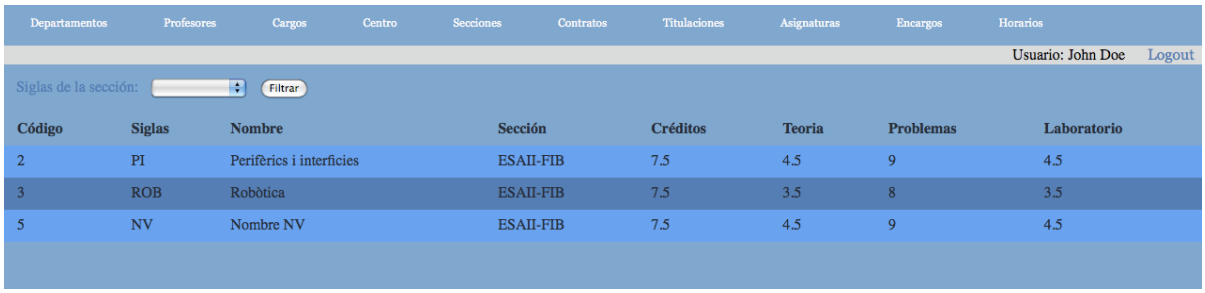


Figura 22 Vista de asignaturas de la sección ESAII-FIB

Como se puede observar, en este caso se muestran las diferentes propiedades de una asignatura, su código, sus siglas, su nombre completo, la sección que la imparte, los créditos que vale y los puntos de teoría, problemas y laboratorio que suponen para los profesores que las imparten.

Además también se sigue mostrando el menú desplegable, que permite seleccionar otra sección, para consultar las asignaturas de otra sección.

10.- Vista de encargos

La vista de encargos muestra la información de los encargos de asignaturas recibidos por una sección en un cuatrimestre, por lo que esta vista ofrece dos menús desplegables mediante los cuales se puede elegir la combinación de estos dos parámetros que se desee.

En esta vista se muestra una fila de información por asignatura encargada a la sección seleccionad en el cuatrimestre seleccionado. De cada encargo se muestran el centro que realiza el encargo, la titulación para la que se realiza el encargo, el cuatrimestre, la asignatura encargada y el número de grupos de problemas, laboratorio y teoría de la asignatura encargados.

La información de los encargos se utiliza para generar los archivos Excel que se pueden descargar para realizar las asignaciones de horas lectivas a profesores.

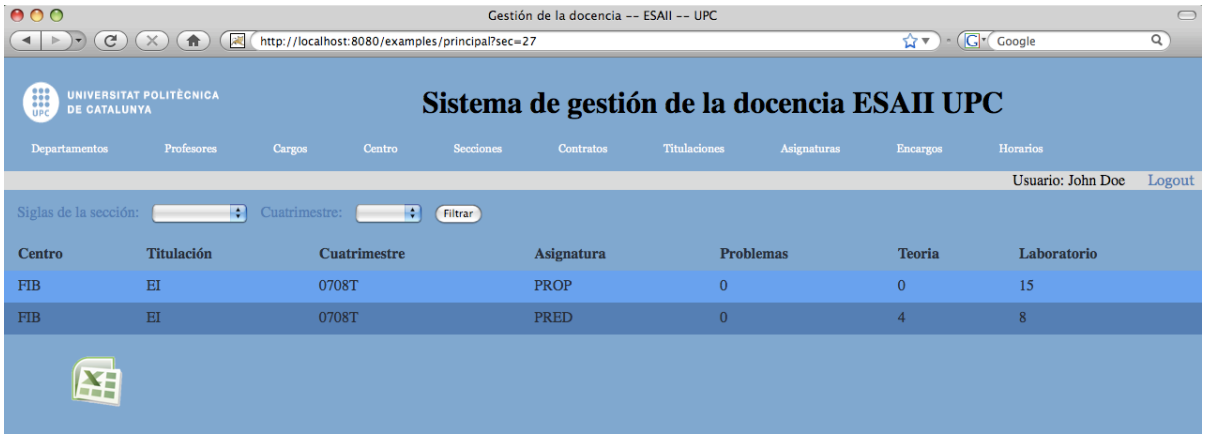


Figura 23 Vista de encargos

10.1.- Subida de encargos mediante archivos xls

Los encargos se introducen en el sistema mediante la carga de archivos xls. Al pinchar en el icono que aparece en la parte inferior de la figura 23, se muestra una ventana emergente en la que hay un formulario que muestra una ventana de selección de ficheros que permite elegir el archivo a cargar.

10.2.- Estructura de los archivos xls para subir encargos

Los archivos xls que se usen para subir encargos al sistema, deben seguir la estructura del fragmento que se muestra en la siguiente tabla.

0708T	LSI-FIB	EI	FIB					
codigo	siglas	nombre	puntosT	puntosP	puntosL	gruposT	gruposP	gruposL
6	PROP	Projecte de programació	0	0	7.5	0	0	15

La primera fila de la hoja de cálculo debe contener el cuatrimestre para el que se realiza el encargo, la sección que debe realizar el encargo, la titulación para la que se realizan esas asignaturas y el centro que ha realizado el encargo.

La siguiente fila es simplemente una cabecera, para saber en que orden se han de poner los datos, si se desea se puede dejar en blanco.

Por último, a partir de la tercera fila, incluyéndola, cada fila constituye un encargo de una asignatura, donde se especifican la información completa de la asignatura y el número de grupos que se encargan. Tal y como se muestra en la tabla del ejemplo.

11.- Vista de horarios

En la vista de horarios se pueden consultar las horas de clase que imparte un profesor en un cuatrimestre. El profesor y el cuatrimestre se seleccionan mediante dos menús desplegables, de la misma manera que la sección y el cuatrimestre en el caso de los encargos.

Tal y como se muestra en la figura 24, se muestra una fila de información para cada hora de clase que el profesor imparte, indicando la hora de inicio y fin de la clase, el día en que se imparte esa clase, la asignatura, el grupo y el centro y el aula donde se imparte la asignatura.

Figura 24 Vista de horarios

11.1.- Descarga de horarios mediante archivos xls

Los horarios de un profesor también se pueden descargar en formato xls. Clicando en el icono que aparece en la parte inferior de la figura 24. Aparece la ventana que se puede observar en la figura 25.

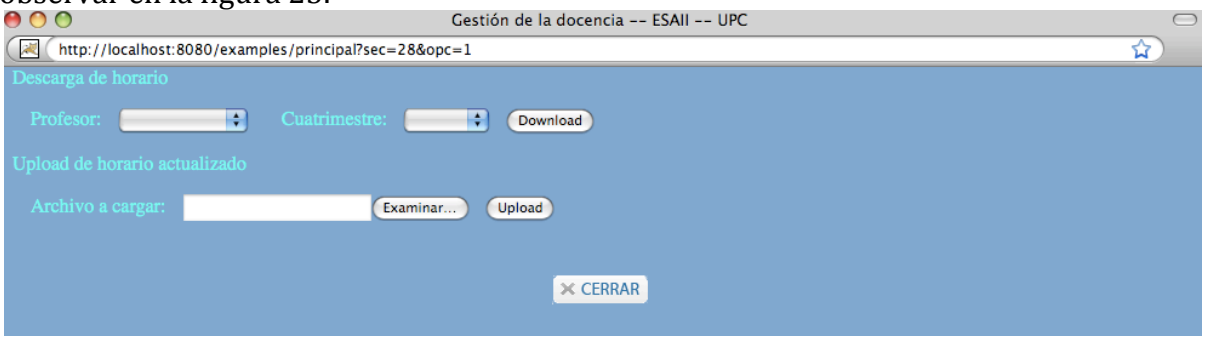


Figura 25 Pop-up de interacción de archivos xls en la vista de horarios

En esta ventana aparecen dos formularios, el primero de ellos posee dos menús desplegables, que sirven para seleccionar el profesor y el cuatrimestre del que se quiere descargar el horario en formato xls.

La estructura de este archivo es similar a la mostrada en la siguiente tabla.

Profesor	John Doe	12345678	0708T
4	II	T1	
viernes	15:00:00	17:00:00	A4001
5	NV	L10	
5	NV	L12	
lunes	10:00:00	12:00:00	A5202

La primera fila contiene el nombre y DNI del profesor y el cuatrimestre seleccionado. Tras esta primera fila aparece la estructura de horarios.

Esta estructura está organizada de la siguiente manera. Primero aparece una línea para cada grupo en el que el profesor tiene asignado algún punto, en el ejemplo las líneas azules. Debajo de esa línea se colocan las distintas horas de clase que ese profesor imparte en esa asignatura, indicando el día, las horas de inicio y fin y el aula, en este ejemplo son las líneas naranjas.

Este formato se repite para todos los grupos en los que el profesor tiene alguna asignación. Si para un grupo no hay definida ninguna clase en la siguiente línea se muestra el siguiente grupo.

11.2.- Subida de horarios mediante archivos xls

El segundo formulario de la ventana que aparece en la figura 25 permite seleccionar un archivo para subirlo y cargar los horarios, allí definidos, en el sistema.
Estos archivos siguen el formato descrito en el apartado anterior, por lo que se recomienda, cuando se quiera actualizar un horario, primero descargar ese horario, como se ha descrito anteriormente y modificarlo cumpliendo el formato definido en el apartado anterior.

Se ha de tener en cuenta que un profesor solamente puede modificar sus horarios, por lo que si sube un archivo que intenta modificar los horarios de otro, el sistema no realizará ninguna modificación sobre su configuración y avisará al usuario.

Anexo 2: Bibliografía

[1] Froufe Quintas, A. : *Java 2: Manual de usuario y tutorial*. 4ª edición. Ra-Ma. Madrid, España : 2005. ISBN 84-7897-679-5

[2] Perry B.W.: *Java Servlets & JSP Cookbook.* 1ª edición. O-Reilly. Sebastopol, Estados Unidos de América. 2004. ISBN 978-0-596-00572-6

[3] Eguíluz Pérez, J. *Introducción a XHTML. http://www.librosweb.es/xhtml/pdf/* . 2008

[4] Donald D. Chamberlin. OH 329. *Oral history interview by Philip L. Frana*, 2001, San Jose, California. Charles Babbage Institute, University of Minnesota, Minneapolis. *http://www.cbi.umn.edu/oh/display.phtml?id=317*

[5] http://www.faqs.org/docs/ppbook/c1164.htm

[6] http://www.w3schools.com/css/

[7] http://www.w3schools.com/html/

[8] http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html

[9] http://tomcat.apache.org/

[10] http://www.mysql.com/

[11] http://poi.apache.org/

[12] http://commons.apache.org/fileupload/apidocs/index.html